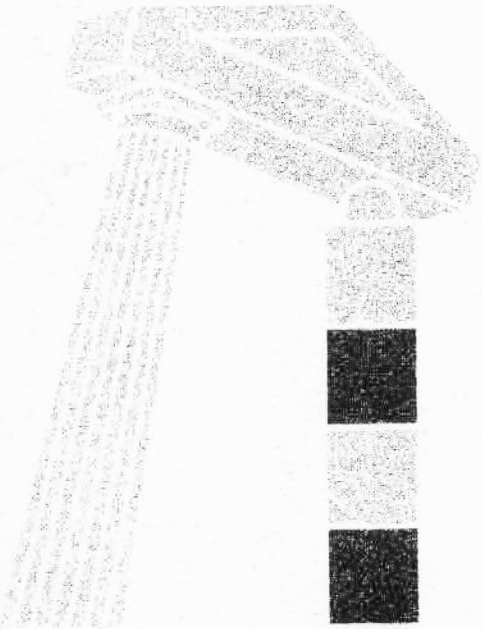
# EWCBR-93

## First European Workshop on Case-Based Reasoning



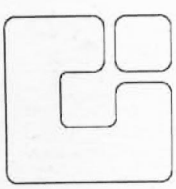
## Posters and Presentations

- Volume I -

**M. M. Richter, S. Wess,**
**K.-D. Althoff, F. Maurer (Eds.)**

1 - 5 November 1993

University of Kaiserslautern (Germany)

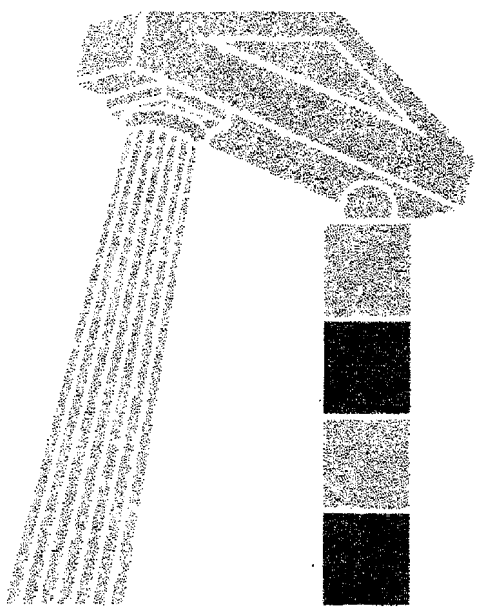


ECCAI


SFB 314

# EWCBR-93

## First European Workshop on Case-Based Reasoning



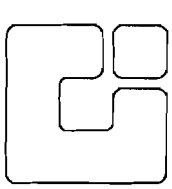
## Posters and Presentations

- Volume I -

**M. M. Richter, S. Wess,**
**K.-D. Althoff, F. Maurer (Eds.)**

1 - 5 November 1993

University of Kaiserslautern (Germany)



ECCAI


SFB 314

# First European Workshop on Case-Based Reasoning (EWCBR) Presentations and Posters Volume I

Michael M. Richter, Stefan Wess,
Klaus-Dieter Althoff, Frank Maurer (Eds.)

SEKI Report SR-93-12 (SFB 314)
Fachbereich Informatik
Universität Kaiserslautern
Postfach 3049
D-67653 Kaiserslautern
Germany

## Program Chair

Prof. Dr. Michael M. Richter
University of Kaiserslautern
Department of Computer Science
P.O. Box 3049
D-67653 Kaiserslautern, Germany

## Program Committee

| | |
|---|---|
| Agnar Aamodt | (Trondheim, Norway) |
| Jaime G. Carbonell | (Pittsburgh, U.S.A.) |
| Thomas Christaller | (Sankt Augustin, Germany) |
| Boi V. Faltings | (Lausanne, Switzerland) |
| Klaus P. Jantke | (Leipzig, Germany) |
| Mark T. Keane | (Dublin, Ireland) |
| Janet L. Kolodner | (Atlanta, U.S.A.) |
| Michel Manago | (Paris, France) |
| Ramon Lopez de Mantaras | (Blanes, Spain) |
| Bernd Neumann | (Hamburg, Germany) |
| Bruce W. Porter | (Austin, U.S.A.) |
| Frank Puppe | (Würzburg, Germany) |
| Lorenza Saitta | (Torino, Italy) |
| Derek Sleeman | (Aberdeen, UK) |
| Gerhard Strube | (Freiburg, Germany) |
| Walter Van de Velde | (Brussels, Belgium) |

## Additional Reviewers

| | |
|---|---|
| Klaus-Dieter Althoff | (Kaiserslautern, Germany) |
| Kerstin Dautenhahn | (Sankt Augustin, Germany) |
| Pete Edwards | (Aberdeen, UK) |
| Wolfgang Gräther | (Sankt Augustin, Germany) |
| Thomas Hemmann | (Sankt Augustin, Germany) |
| Beatriz Lopez | (Blanes, Spain) |
| Frank Maurer | (Kaiserslautern, Germany) |
| Rüdiger Oehlmann | (Aberdeen, UK) |
| Enric Plaza | (Blanes, Spain) |
| Luigi Portinale | (Torino, Italy) |
| Barbara Schmidt-Belz | (Sankt Augustin, Germany) |
| Jürgen Walter | (Sankt Augustin, Germany) |
| Stefan Wess | (Kaiserslautern, Germany) |

## Organizing Committee

Stefan Wess, Klaus-Dieter Althoff, Frank Maurer
University of Kaiserslautern
Department of Computer Science
P.O. Box 3049
D-67653 Kaiserslautern, Germany
Tel.: +49 631 205 3360 (3362,3356,3363)
Fax: +49 631 205 3357
email: ewcbr@informatik.uni-kl.de

## Organization

EWCBR-93 is organized by
the Expert System Section of the German Society for Computer Science (GI),
the Special Interest Group on Case-Based Reasoning (AK-CBR),

in cooperation with
the European Coordinating Committee for AI (ECCAI),
the German Chapter of the Association for Computing Machinery (ACM),
the Computer Science Department of the University of Kaiserslautern,
the German Special Research Investigation on Artificial Intelligence
and Knowledge-Based Systems (SFB 314) of the DFG,
the German Research Center on Artificial Intelligence (DFKI), Kaiserslautern

# Preface

The *First European Workshop on Case-Based Reasoning* (EWCBR-93) is aimed at researchers and practitioners interested in the methodological progress and the extensions of the areas of application of Case-Based Reasoning. Case-Based Reasoning is a topic which becomes more and more important and has raised considerable interest recently. It supports knowledge acquisition and problem solving, and it is related to key words like machine learning, analogy, cognitive modeling, similarity, information retrieval among others. Although case-based reasoning has a well defined place within AI-related conferences, we felt that the topic deserves a workshop on its own also in Europe, as there have been such events in the US.

The program committee accepted 21 submissions for presentation at the workshop and about 50 submissions for the poster sessions. This volume contains all these extended abstracts. The scientific program also includes four invited talks, system demonstrations as well as one panel discussion. An overview on Case-Based Reasoning as well as the presentation of commercial CBR systems is scheduled for the first day.

An overview on Case-Based Reasoning is given by Agnar Aamodt (University of Trondheim, Norway) and Enric Plaza (CEAB-CSIC, Spain).

Invited talks cover important aspects of Case-Based Reasoning:

Janet L. Kolodner (Georgia Institute of Technology, U.S.A.):
*Making Computers Creative - A Case-Based Approach.*

Katharina Morik (University of Dortmund, Germany):
*A Case for Inductive Learning.*

Mark T. Keane (Trinity College, Ireland):
*Analogical Asides on Case-Based Reasoning.*

Manuela Veloso (Carnegie Mellon University, U.S.A.):
*Analogical/Case-Based Reasoning in General Problem Solving.*

We thank all who submitted their papers. We are most grateful to the members of the program committee for carrying out the difficult task of paper selection. It was the general feeling that we had an unusual responsibility because this is the first workshop on the topic in Europe and will most likely be a milestone for the future development of this area of research.

The forthcoming proceedings of the workshop will contain the final versions of a selection of long papers.

*Michael M. Richter*
(program chair)

*Stefan Wess, Klaus-Dieter Althoff, Frank Maurer*
(organizing committee)

## Wednesday, Nov. 3rd

9.00h Katharina Morik (University of Dortmund, Germany)
*A Case for Inductive Learning (Invited Talk)*

11.00h **Session 3:**

Kuniaki Uehara, Masayuki Tanizawa, Sadao Maekawa, Japan:
*PBL: Prototype-Based Learning Algorithm*

Michel Manago, Klaus-Dieter Althoff, Noel Conruyt, Frank Maurer, Ralph Traphöner, Stefan Wess, France:
*Induction and Reasoning from Cases*

Yoshio Nakatani, David Israel, Japan:
*Tuning Rules by Cases*

14.00h **Session 4:**

Petri Myllymäki, Henry Tirri, Finland:
*Massively Parallel Case-Based Reasoning with Probabilistic Similarity Metrics*

Scott O'Hara, Bipin Indurkhya, USA:
*Incorporating (Re)-Interpretation in Case-Based Reasoning*

16.00h **Poster Presentation (part 2)**

19.30h **Working Groups**

**Adaptation and Analogy**
Dietmar Janetzko (University of Freiburg);
Mark T. Keane (Trinity College, Dublin).

**Integrated Problem Solving and Learning Architectures**
Agnar Aamodt (University of Trondheim);
Klaus-Dieter Althoff (University of Kaiserslautern).

**Knowledge Engineering and Case-Based Reasoning**
Frank Maurer (University of Kaiserslautern);
Gerhard Strube (University of Freiburg).

**Retrieval, Similarity and Indexing**
Klaus P. Jantke, (HTWK Leipzig);
Michael M. Richter (University of Kaiserslautern).

**Case-Based Decision Support/Case-Based Diagnosis**
Brigitte Bartsch-Spörl (BSR Consulting, Munich)
Michel Manago (AcknoSoft, Paris)

**Case-Based Design/Case-Based Planning**
Ian Smith (Swiss Federal Institute of Technology, Lausanne)
Manuela Veloso (Carnegie Mellon University, Pittsburgh)

## Thursday, Nov. 4th

09.00h Mark T. Keane (Trinity College, Dublin, Ireland)
*Analogical Asides on Case-Based Reasoning (Invited Talk)*

11.00h **Session 5:**

Ralph Bergmann, Gerd Pews, Germany:
*Explanation-based Similarity for Case Retrieval and Adaptation and its Application to Diagnosis and Planning Tasks*

Luigi Portinale, Pietro Torasso, Carlo Ortalda, Antonio Giardino, Italy:
*Using Case-Based Reasoning to Focus Model-Based Diagnostic Problem Solving*

Barry Smyth, Mark T. Keane, Ireland:
*Retrieving Adaptable Cases: The Role of Adaptation Knowledge in Case Retrieval*

14.00h **Session 6:**

Gerd Kamp, Germany:
*Integrating Semantic Structure and Technical Documentation in Case-Based Service Support Systems*

David Macchion, Dinh-Phuoc Vo, France:
*A Hybrid KBS for Technical Diagnosis Learning and Assistance*

Carol Bradburn, John Zeleznikow, Australia:
*The Application of Case Based Reasoning to the Tasks of Health Care Planning*

16.00h **Session 7:**

Mike G. Brown, UK:
*An Under-Lying Memory Model to Support Case Retrieval*

Uri J. Schild, Yaakov Kerner, Israel:
*Multiple Explanation Patterns*

Kevin D. Ashley, Vincent Aleven, USA:
*Using Logic to Reason with Cases*

18.00h **Panel Discussion**
*Case-Based Reasoning a new Research Paradigm?*
*- Positioning Case-Based Reasoning -*
Chair: Michel Manago

## Friday, Nov. 5th

09.00h Manuela Veloso (Carnegie Mellon University, Pittsburgh, U.S.A.)
*Analogical/Case-Based Reasoning in General Problem Solving (Invited Talk)*

11.00h **Discussion: Case-Based Reasoning in Europe**
Chair: Michael M. Richter

# Program Schedule

## Monday, Nov. 1st

09.00h Agnar Aamodt (University of Trondheim, Norway)
*Case-based Reasoning: An Overview (part 1)*

11.00h Agnar Aamodt (University of Trondheim, Norway)
*Case-based Reasoning: An Overview (part 2)*

14.00h Enric Plaza (CEAB-CSIC, Spain)
*Case-based Reasoning: An Overview (part 3)*

16.00h **Presentation of Commercial CBR Tools**

## Tuesday, Nov. 2nd

09.00h Janet L. Kolodner (Georgia Institute of Technology, U.S.A.)
*Making Computers Creative: A Case-Based Approach (Invited talk)*

11.00h **Session 1:**

Agnar Aamodt, Norway:
*Explanation-Driven Retrieval, Reuse and Learning of Cases*

Dean Allemang, Switzerland:
*Case-Based Reasoning and Task-Specific Architectures*

Enric Plaza, Josep-Lluis Arcos, Spain:
*A Reflective Architecture for Integrated Memory-based Learning and Reasoning*

14.00h **Session 2:**

Kefeng Hua, Ian Smith, Boi Faltings, Switzerland:
*Integrated Case-Based Building Design*

Angi Voss, Germany:
*How to interpret design sketches? - Different approaches to similarity in FABEL*

Amedeo Napoli, Jean Lieber, France:
*Finding Strategies in Organic Synthesis Planning with Case-Based Reasoning*

16.00h **Poster Presentation (part 1)**

19.30h **Demonstration of Participant CBR Systems**

# Contents

# Chapter 1

# Retrieval, Similarity and Indexing

# ANAIS: A Case-Based Reasoning System in an Problem Solving Environment

Nathalie Beaubouchet
Unité de recherche INRIA Rhône-Alpes
LIFIA 46 avenue Félix Viallet
38031 Grenoble Cedex, France
Email: Nathalie.Beaubouchet@imag.fr

## Introduction

In case-based reasoning approach, new problems are solved using past solutions of previously solved problems. Analogy is based on an extraction of solved problems, and the most important difficulty is finding significant aspects shared by the solved problems and the new problem [Car86]. Representing a case not only by its characteristics but by memorizing a trace of the problem solving reasoning for this case extends the scope of case-based reasoning from problem solving to knowledge acquisition. In ANAIS (ANAlogical Intelligent System), a case is a set of characteristics organized in a hierarchy, and also an instantiated task network which represents the reasoning. Two phasis are used to select the most similar case. In the first one, a selection of some cases is based on the characteristics, and in the second one, a retrieval of the most similar case is based on a matching algorithm of reasonings. Although this study is independant of any domain, an application is implemented in electromyography. The electromyography is a diagnosis medical technique, its results are used to predict muscular or nervous diseases.

## 1 The first phase: a preselection

In order to retrieve cases similar to the new problem from memory, the first phase has to select cases with the same characteristics before matching the reasoning. The main difficulty encountered is to formulate the characteristics of a problem. In electromyography, few characteristics are avaible, such as: general disease, suspected diagnostic to be confirmed, and some important symptoms. Because of small number of characteristics, the reasoning matching phase is important to refine the set of cases selected by the first phase.

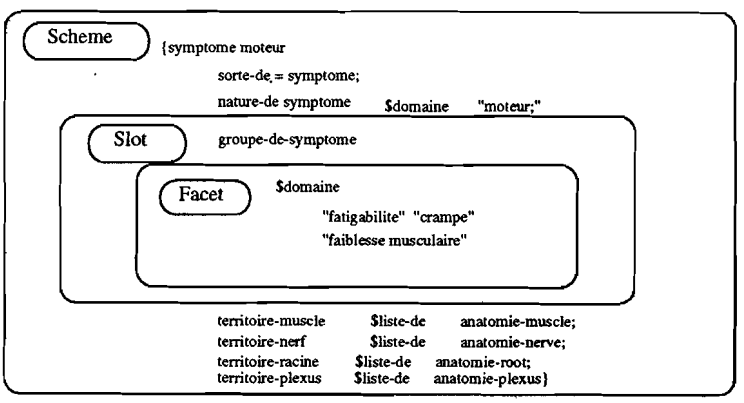



Figure 1: A scheme in SHIRKA

The characteristics are stored in SHIRKA [RU91] representation knowledge model. A Shirka entity describes a class of objects and its instances. An object called a "scheme", is

defined by its slots, each slot having several facets. A facet can be a list of possible values, some constraints on the slot value or can reference another scheme (Figure 1). The classes are organized in a specialisation hierarchy with inheritance mechanism, which means that a class inherits slots from its super-classes. The inherited slots from the super-classes also inherit their constraints and a class can have some additionals slots which are its own slots.

In ANAIS, the slots of a class represent the characteristics of a case, except one slot which is a list pointing on cases sharing these characteristics. The root class has been defined with one shared slot : the slot "cas" in wich are founded all the cases attached to the class. All the existing characteristics of the domain are supposed to be stored in a hierarchy (Figure 2). Adding some new characteristics means to modify the hierarchy, it can be the moving of classes, the creation of a new slot in a class or a new class. Coherence problems can appear and the coherence maintenance is also a preoccupation in the laboratory [Cap93].

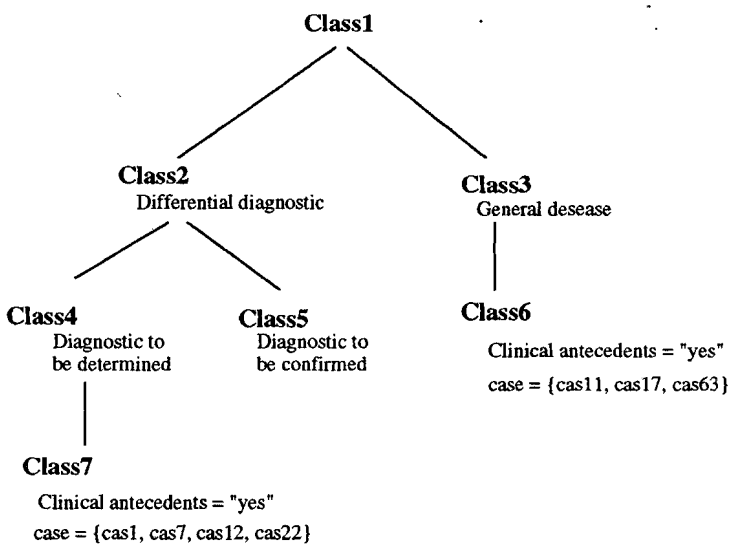



Figure 2: The hierarchy of characteristics

When the user tries to solve a new problem, he provides some of its characteristics. For example, in medical diagnosis, these characteristics can be probable diagnostic already known, general desease, clinical antecedents or muscular desease. With these given characteristics and those defined in the hierarchy, some cases can be selected. This extraction relies on the SHIRKA classification algorithm. This one allows a given instance with some slot values attached to the root class, to determine all possible classes the instance could be attached to [1]. So, the user creates a new instance of the root class, he provides some characteristics (slot values), and the classification algorithm finds all possible classes of the instance. With the slot "cas" of each possible class, a set of cases is built. Thus, the matching reasoning phase is relieved with this reduced number of cases.

# 2 Representation of problem solving reasoning

The reasoning representation of a case takes place in the problem solving environment SCAI (Scientific Computing with Artificial Intelligence) [PR91]. In this formalism, a task is modeled as a class with slots describing its inputs and outputs, and can be decomposed

[1] For more information about the classification algorithm see [MRU90] or [Mar93]

into sub-tasks until elementary tasks, corresponding to a decomposition of a complex problem into sub-problems. This model had been built onto the SHIRKA representation system, thus this phasis is naturally integrated with the previous one. The reasoning trace is described by a hierarchical network of instantiated tasks.

A complex task can be defined by sequential sub-tasks or choice sub-tasks (automatic choice or interactive choice). An elementary task references a method to be executed. All types of tasks can be specialized, allowing a context adaptation. Iteration and recursion are defined explicitly by a recall of a task in one of its subtasks (Figure 3).

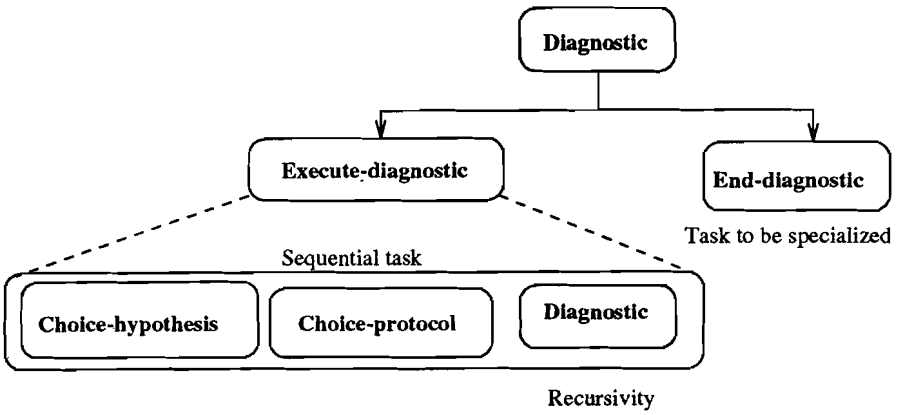



Figure 3: Representation of tasks in SCAI

The first assumption is that there is a task base in which the user can choose a task and execute it automatically or decompose it himself. This base allows to solve some problems in a specific domain. When the user tries to solve a problem, sometimes he realizes that he cannot completely solve it with the actual base, even if some tasks are usefull. With the usefull tasks indicated by the user, ANAIS provides the similar cases, in order to allow the user to extend the knowledge base of tasks to solve the new problem. In this way, the task base is improved covering more and more problems. Another way is when the user is lost in the solving of a problem, a similar case can help him. The user can also have found a new way to solve a problem and he wants to extend the knowledge base. In our problem solving environnement, several task bases can be loaded, and the user can pick some tasks in the differerent bases to solve a new problem. Providing similar cases in each base can help the user to build other complex tasks in order to cover the solving of the new problem.

In each of these instances, the user wants to solve a new problem and he partially gives the reasoning to achieve it. Thus, he describes a set of instantiated tasks which are decomposed or not. So, the description of a new problem to be solved consists in two parts, the characteristics of the problem, and some parts of instantiated networks corresponding to a partial resolution of the problem. The extracted cases from the first phase are compared with partial descriptions corresponding to a partial reasoning in order to obtain the most similar reasoning. Determining the most similar case in the second phase needs to match instantiated task networks and to compute their similarity.

# 3 Second phase: case evaluation and similarity

In the environment problem solving, there are two modes of execution, a free one and a guided one. In the guided mode, the user gives the principal task and the inference mechanism system automatically decomposes and solves the problem according to the

knowledge base decomposition of the task. In the free mode, the user can choose many tasks and solve them independently. In this mode, a partial reasoning can be expressed.

Matching these partial descriptions with past stored reasoning allows the retrieval of the most similar solution, and allows the user to complete his reasoning with the similar case and extend the task base. The partial resolution of the new problem is memorized with the history of executed tasks. This history is a set of executed tasks and is represented in a LISP list in which all sublists are subtasks of a high level task. The reasonings of selected cases is a list of tasks, also in LISP formalism.

In the matching function, the input is a list corresponding to the reasoning of a case, and a partial reasoning described by the user. The output of a first algorithm is a list of all tasks shared by the inputs. The evaluation of these shared tasks depends on the task importance and on the generality or specificity of these tasks. Some tasks are independent of the case, for example, in electromyography, the diagnostic always begins with anterior face and extremity examinations. These tasks have not to be taken into account for the similarity calculus.

Then, some equivalence rules are necessary to determine the similarity of reasonings. For example, if a task `t1` is not explicitly in one of the input, but if chained subtasks executing completely `t1` appear, these subtasks have to be considered equivalent to `t1` and vice-versa. Some simplification rules are also necessary, like: cut the recursive-task associated tree, cut the terminal-task associated procedure, cut the control task and transformations are to be used to identify equivalent sequences. Some rules are independent of the domain and others are expressed according to electromyography.

Before the similarity assessment, significance are to be associated to all executed tasks. Now, a significance task depends on the domain and may be also on the specific case. To avoid to the user giving these significance values, we have some rules for the relative importance of tasks. For example, more a task has level of subtasks or has direct subtasks, more it is an important one and recursive tasks are the most important ones. Actually, a structural similarity assessment defined by [Bis92] is studied to fit as well as possible the human reasoning in the domain.

# Issues

In electromyography, which is a set of techniques allowing the diagnostic of nervous or muscular diseases, a system has been developped [ZVC92]. However, this system does not take all the reasoning into account. The behavior of the system is like a beginner, executing all the possible examinations for a disease hypothesis. Giving case-based reasoning to the system can allow the examination protocol to be optimal. In addition of the case representation and the retrieval process of a similar case, an objective is to improve the knowledge base.

In the matching of reasonings, numerous equivalence and simplification rules are necessary. These rules have to be independant of the application domain and are based on the environment problem solving definition. Otherwise, the task significance can depend on the considered domain. In issue, the generalization of similar problems into a generic reasoning model can be an extension of case-based reasoning. The first advantage is to relieve the case memory, in substituting the similar cases by their generic models. The second advantage is to improve knowledge acquisition by using these generic models for the problem reasoning expression.

# References


[Bea93] Nathalie Beaubouche. Anais : Un système de raisonnement à partir de cas dans un environnement de résolution de problèmes. In *Journées VOLCAN-IA 93*, pages 245–260, Clermont-Ferrand (France), March 1993.

[Bis92] Gilles Bisson. Learning in FOL with a similarity measure. In *AAAI*, pages 82–87, 1992.

[Cap93] Cécile Capponi. Classification des classes par les types. In *2nd journées représentation par objets (RPO)*, pages 215–224, La Grande Motte (France), June 1993.

[Car86] Jaime G. Carbonell. Derivational analogy : a theory of reconstructive problem solving and expertise acquisition. In *Machine Learning, An artificial intelligence approach*, pages 371–391. Morgan Kaufman, 1986. Vol 2.

[Mar93] Olga Marino. *Classification multi-points de vue dans un environnement objet.* PhD thesis, Université Joseph Fourier, October 1993.

[MRU90] O. Marino, F. Rechenmann, and P. Uvietta. Multiple perspectives and classification mechanism in object-oriented representation. In *ECAI90*, pages 425–430, 1990.

[PR91] Thierry Poncabarré and François Rechenmann. SCAI: un environnement de développement de systèmes à base de connaissances en calcul scientifique et technique. In Hermès, editor, *Convention intelligence artificielle*, Paris, january 1991.

[RU91] François Rechenmann and Patrice Uvietta. Shirka, an object-centered knowledge base management system. In *Artificial Intelligence in Numerical and Symbolic Simulation*, Lyon, 1991. ALEAS Publ.

[ZVC92] D. Ziébelin, A. Vila, and F. Cruypenninck. Automatic reasoning and explanation in knowledge based system for emg diagnotics. In *IX International congress of electromyography and clinical neurophysiology*, Jerusalem, Israel, 1992.

# A Similarity Metric for Retrieval of Cases Imperfectly Described and Explained

**Carlos Bento Ernesto Costa**
bento@alma.uc.pt ernesto@moebius.uc.pt
Laboratório de Informática e Sistemas - Univ. de Coimbra
Quinta da Boavista, lote 1, 1°
3000 Coimbra - PORTUGAL
telef.: +351 39 402479 fax: +351 39 701266

**Abstract.** In this paper we present a quantitative similarity metric for retrieval of past cases imperfectly described and explained. We introduce the concepts of matching situation and situation snippet which are used in our metric.

We describe CLASH, a Case-Based Reasoning Expert System implemented in PROLOG, which applies this metric. The results provided by CLASH are compared with another system based on a different metric for case retrieval.

## 1. Introduction

The power of a Case-Based Reasoning (CBR) System [10, 5, 3] is greatly determined by its capability to retrieve the relevant cases for prediction of the new outcome. The retrieval process involves indexing cases.

A nearest neighbour algorithm for case retrieval, described by Duda *et al.* [2], searches through every case in memory, applies a similarity metric and returns the case (or k cases) with the past situation most similar to the new situation. This similarity metric counts the number of facts that the past and the new situations have in common.

Two other systems CYRUS [4] and UNIMEM [6] index cases by facts in the past situation that are predictive of other facts in the outcome. Predictiveness of the facts is determined by some correlation calculations. This has some drawbacks, specially when calculations are performed on a small data set [8].

The combination of nearest neighbour and knowledge-guided techniques led to the development of hybrid systems joining CBR and Explanation-Based Learning (EBL) techniques [7]. These systems use domain knowledge for constructing explanations of why a situation had a specific outcome in the past. These explanations are necessary to judge the relevance for future retrieval of the facts describing a past situation. This approach was followed by Cain *et al.* [1]. They use a CBR+EBL similarity metric in which case explanations influence but do not determine the relevance assigned to past situation facts.

Past case explanations are subject to imperfections. These are related to the absence of a perfect theory on the domain and to imperfections in the past situation description (complete proof trees can not be constructed when the situation description lives out some relevant facts ).

We report three kinds of imperfections in explanations: (1) broken explanations; (2) partial explanations; (3) incomplete set of explanations. Broken and partial explanations, not considered in Cain's metric, are discussed in this paper.

In section 2, we make a brief overview of the similarity metric for retrieval proposed by Cain *et al.* and report four limitations in this metric. In section 3, we describe a new metric that overcomes the drawbacks reported before. In section 4, we present CLASH, a Case-Based System, implemented in PROLOG, for evaluation of the highway code offences that where in the origin of car accidents. This system uses our metric. We compare the ability of CLASH to retrieve relevant past cases with a system that uses the Cain similarity metric. Finally, in section 5, we make some comments concerning the advantages of our approach.

## 2. A Brief Overview of Cain *et al.* Similarity Metric

In Cain *et al.* approach a case is composed by a set of facts that represent a past situation (PS), another set of facts that represent an outcome (OUTC) and a set of explanations (EXPS) of why the situation had such an outcome. A new situation (NS) is also represented by a set of facts.

Cain *et al.* use a parameterized similarity function influenced by the explanations produced by the domain

theory for each case in memory:

$$\frac{\alpha \sum_{i=1}^{n} sim(f_i, f_i') + \beta \sum_{i=1}^{n} relevance(f_i) * sim(f_i, f_i')}{\alpha n + \beta \sum_{i=1}^{n} relevance(f_i)} \quad (1)$$

where :

$$sim(fi, fi') = \begin{cases} 1 & \text{if } fi = fi' \\ 0 & \text{if } fi \neq fi' \end{cases} ; \qquad relevance(fi) = \begin{cases} 1 & \text{if } fi \text{ is relevant} \\ 0 & \text{if } fi \text{ is irrelevant} \end{cases} ;$$

$f_i$ is a fact in the past situation, $f_i'$ is a fact in the new situation. A fact $f_i$ is relevant if it is included in an explanation for the past case. The $\alpha$ parameter represents the weight of a match between any fact in the past situation and a fact in the new situation. Parameter $\beta$ represents the additional weight of a match between a relevant fact in the past situation and the same fact in the new situation.

If $\beta$ is set to zero then the evaluation function ignores the relevance of facts and a pure Similarity-Based match is performed. With positive values for $\alpha$ and $\beta$ a CBR+EBL based metric is performed.

Although this metric has produced interesting results when compared with a nearest neighbour based retrieval or with a pure knowledge-guided retrieval the following points weaken Cains *et al.* approach:

(1) It assumes all explanations are complete.
As it is accepted that past situation descriptions and domain theory are both imperfect it is expected to have imperfect explanations in cases.
Facts in the past situation that are relevant for imperfect explanations must have a different treatment from those that are relevant for complete explanations.

(2) It assigns the same relevance to a fact, independently of belonging to a small or a large set of facts that as a whole influences or determines an outcome fact.
It is expected that the unmatching of a fact from a set with few facts that influences or determines an outcome fact is more harmful than the unmatching of a fact from a set with many facts. So, it must be assigned a higher relevance to a fact of the first type than to one of the second type.

(3) It does not discriminate between a fact that is relevant for one explanation and one that is relevant for several explanations.
It is sound to assign a higher relevance to a fact which influences several outcome facts than to one that influences only one outcome fact.

(4) It does not discriminate between two cases, one with a complete set of explanations and another with an empty set of explanations .
As past situations may be imperfectly described it is sensible to prefer cases that are explained over cases that are not. In a completely explained case it is known which facts in the past situation description influence or determine the outcome. In a case with an empty set of explanations this is unknown.

In the next section we describe a similarity metric that does not suffer from these limitations.

## 3. An Alternative Approach

In our approach a case is composed of a past situation, an outcome and a set of explanations of why the situation had such outcome. We consider the three kinds of explanation imperfections described before: (1) broken explanations; (2) partial explanations; (3) incomplete set of explanations.

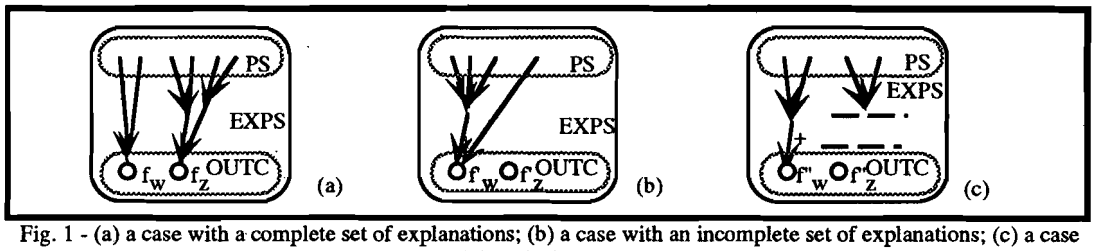



Fig. 1 - (a) a case with a complete set of explanations; (b) a case with an incomplete set of explanations; (c) a case with a partial and a broken explanation.

A broken explanation is one in which there is a gap between the proof tree and the case outcome (e.g., In fig. 1c the second proof tree from the left). A partial explanation is one whose proof tree omits some branches. Branches labeled with a '+' represent a step in which some are absent) (e.g., In fig. 1c the second step in the proof tree on the left). In a case with an incomplete set of explanations some outcome facts are not explained and so are not end of a proof tree (e.g., The cases represented in fig.s 1b and 1c. Facts $f'_z$ and $f''_z$ in the outcome are not end of a proof tree).

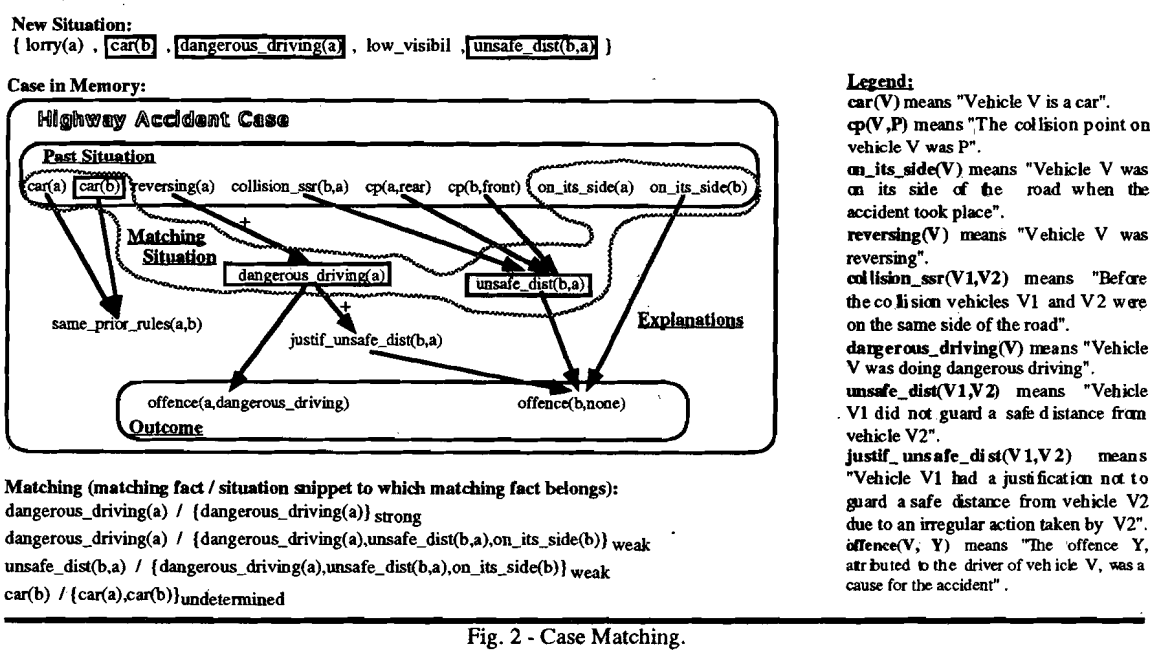



Fig. 2 - Case Matching.

Case indexing involves the concepts of past situation, matching situation, strong, weak, and undetermined situation snippet. A past situation represents a problem or event in the past that had the outcome described in a case. A matching situation is a situation that is obtained by going down (from the past situation to the outcome) in one or more explanation trees in order to gather the maximum number of facts that match the new situation (e.g., In fig. 2, the matching situation {car(a), car(b), dangerous_driving(a), unsafe_dist(b,a), on_its_side(a), on_its_side(b)}).

In our approach, a past or matching situation is seen as composed by a set of situation pieces called situation snippets (in analogy with "case snippets" from Redmond [9]). A situation snippet is a set of facts that are the leaves of a proof tree (the premises of an outcome fact). Depending on the proof tree being complete, partial or broken the situation snippet is strong (e.g., In fig. 2, {dangerous_driving(a)} concerning to 'offence(a,dangerous_driving)'), weak (e.g., In fig. 2, {dangerous_driving(a), unsafe_dist(b,a), on_its_side(b)} concerning to 'offence(b,none)'), or undetermined (e.g., In fig. 2, {car(a), car(b))} concerning to 'same_prior_rules(a,b)'). The situation snippets of a situation are not necessarily disjoined sets of facts (as is the case for the examples of strong and weak situation snippets described above). A fact in a situation that does not belong to any proof tree is a single fact undetermined situation snippet (e.g., In fig. 2, {on_its_side(a)}). If a case has a complete set of complete explanations (proof trees) then the undetermined situation snippets become irrelevant.

Matching between a past case and a new situation is represented by the facts in the matching situation that match a fact in the new situation. Each matched fact has information about the situation snippet to which it belongs. Figure 2 provides an example of a matching between a case on highway accident interpretation and a new situation (more detailed information on this domain is given in the next section). In this example the matching situation is {car(a), car(b), dangerous_driving(a), unsafe_dist(b,a), on_its_side(a), on_its_side(b)}. The matching facts are 'car(b)' which belongs to the undetermined situation snippet {car(a), car(b)}, 'dangerous_driving(a)' which belongs to the strong situation snippet {dangerous_driving(a)} and to the weak situation snippet {dangerous_driving(a), unsafe_dist(b,a), on_its_side(b)}, and 'unsafe_dist(b,a)' which belongs to the weak situation snippet {dangerous_driving(a),unsafe_dist(b,a),on_its_side(b)}.

For case retrieval we propose a similarity metric composed of three terms:

$$\kappa \sum_{i=1}^{k} relev(f_i, SS_u)\, sim(f_i, f_i') + \lambda \sum_{i=1}^{r} relev(f_i, SS_w)\, sim(f_i, f_i') + \mu \sum_{i=1}^{t} relev(f_i, SS_s)\, sim(f_i, f_i') \quad (2)$$

with $f_i$ a fact in the matching situation; $f_i'$ a fact in the new situation; $SS_u$, $SS_w$ and $SS_s$ means, respectively, undetermined weak and strong situation snippets;

$$relev(f_i, SS_{type}) = \frac{1}{\text{"cardinal of the } SS_{type} \text{ set to which } f_i \text{ belongs"}} \quad (3) \; ; \quad \text{and}$$

$$sim(fi, fi') = \begin{cases} 1 & \text{if } fi = fi' \\ 0 & \text{if } fi \neq fi' \end{cases}$$

Constants k, r, t are, respectively, the number of occurrences of the matching situation facts in undetermined, weak and strong situation snippets (remember that the situation snippets are not necessarily disjoint sets).

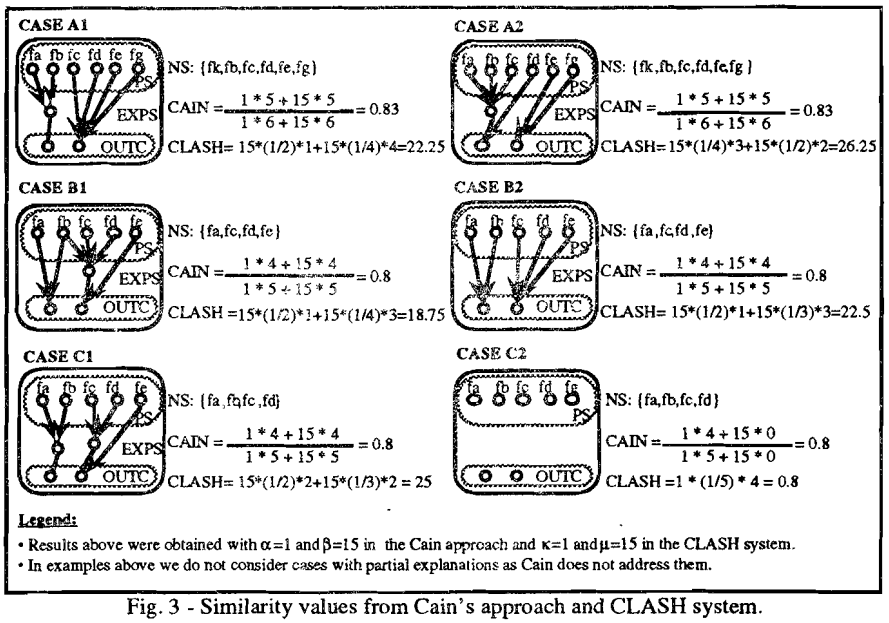



Fig. 3 - Similarity values from Cain's approach and CLASH system.

The ways in which this metric overcomes the limitations pointed to Cain *et al.* approach are reported below:

(1) It discriminates between facts relevant to broken, partial and complete explanations.

The first term in our metric accounts for those matching facts whose influence in the outcome is unknown (matching facts belonging to undetermined situation snippets). The second term respects to matching facts that belong to sets of facts that influence outcome facts but are not sufficient to determine them (matching facts belonging to weak situation snippets). The third term is built by the facts that belong to sets of facts that determine outcome facts (matching facts belonging to strong situation snippets). Parameters $\kappa$, $\lambda$ and $\mu$ represent the weight assigned to the three kinds of matching facts.

(2) It assigns relevance to each matching fact, function of its situation snippet size (see expression (3)).

It is assumed that the relevance of a fact is greater when the number of facts in the situation snippet it belongs is smaller. This is why for CLASH, in fig. 3, the unmatching of $f_a$ is more penalizing for case A1 than for case A2.

Any situation snippet is assigned a unitary relevance value (numerator in expression (3)). This value is divided by the number of facts in it to determine the relevance of a single fact. The reason for this is that any strong, weak or undetermined situation snippet is believed to influence or determine an outcome fact. As the relative importance of outcome facts is unknown it is assumed they all have the same importance. Consequently it is attributed the same relevance to each situation snippet.

(3) It takes into account a matching fact as many times as the number of times it occurs in the situation snippets.

It is sound to assign a higher importance to a fact influencing various outcome facts than to a fact influencing only one outcome fact. This is the reason why for CLASH, in fig. 3, case B2 is less penalized by the unmatching of $f_b$ than case B1.

(4) It assigns a lower similarity value to unexplained cases.

When $\lambda$ and $\mu$ have a value greater than $\kappa$ then a null second or third term in expression (2) is more penalizing for the result than a null first term.

This is why for CLASH, in fig. 3, case C1 has a higher similarity value than case C2.

This similarity metric has been used in CLASH.

## 4. An Example: The CLASH System

Expert inspection of car accidents is a demanding activity for Insurance Companies. To produce faster decisions on compensation for accident losses, they give their costumers a normalized form called "Friendly Accident Declaration". This form is filled when an accident takes place and the drivers agree on the way it occurred. The declaration has seventeen questions of type yes/no about the accident circumstances and a space to draw a sketch of the accident. With the form in their possession, companies are interested in determining (if possible in an automatic way) which driving offences were at the origin of the accident.

CLASH is a prototype of a Case-Based Expert System that has a memory of past accidents. Each case in memory is composed by a past situation that is a transcript of a "Friendly Accident Declaration", an outcome that is a list of offences attributed to the participants in the accident, and a set of explanations of why these offences have been ascribed.

Below we show how a case is represented in the accident library:

**CASE NAME:** case6
**SITUATION:**
[car(a),car(b),reversing(a),collision_ssr(b,a),cp(a,rear),cp(b,front),on_its_side(a),on_its_side(b)]
**OUTCOME:**
[offence(a,dangerous_driving),offence(b,none)]
**EXPLANATIONS:**
[[car(a), car(b)] -> [same_prior_rules(a,b)]
[reversing(a)] ->+ [dangerous_driving(a)]
[collision_ssr(b,a),cp(a,rear),cp(b,front)] -> [unsafe_dist(b,a)]
[dangerous_driving(a)] ->+ [justif_unsafe_dist((b,a)]
[dangerous_driving(a)] -> [offence(a,dangerous_driving)]
[justif_unsafe_dist((b,a),unsafe_dist(b,a),on_its_side(b)] -> [offence(b,none)]
[car(a), car(b)] -> [same_prior_rules(a,b)]]

Case 6 (see also drawing in fig. 2) means:
"A car a was reversing and collided with a car b that was moving on the same side of the road. The collision point on car a was on its rear and on car b on its front. Both cars were moving on their side of the road.
The decision on this accident was that driver of car a committed a dangerous driving offence and driver of car b did not commit any offence.

The explanations were that the reversing of car a influenced the guess that the driver was doing dangerous driving (to have a strong evidence on this offence it would be necessary that the visibility was low, a fact not mentioned in the accident form). The facts that car b was moving on the same side of the road of car a, the collision point on car a was on its rear and on car b on its front determined the evidence that car b did not guard a safe distance from car a. The fact that the driver of car a was doing dangerous driving influenced the guess that the driver of car b could not guard a safe distance. The fact that the driver of car a was doing dangerous driving determined a reason for the accident. The facts that the driver of car b did not guard a safe distance, had a justification not to guard a safe distance from car a and was on his side of the road determined the decision of not assigning any offence to him. The facts that a and b are cars determined that they had to carry the same priority rules".

The case similarity values assigned by CLASH and Cain's metric for three new accidents are printed below. The system based on Cain's metric was initialized with $\alpha$=1 and $\beta$=15. CLASH ran with $\kappa$=1, $\lambda$=7, and $\mu$=15 [1]. The system was setup to return the five cases with the highest similarity value.

**TEST**
<CAIN coefficients> Alpha= 1 Beta= 15 <CLASH coefficients> Kappa= 1 Lambda= 7 Mu= 15

**NEW SITUATION 1** ---- [car(a), car(b), from_the_right(b,a), cp(a, rear), cp(b, front), on_its_side(a), on_its_side(b)]
CAIN Approach: **0.85/case3**, 0.76/case5, 0.66/case1, 0.61/case6, 0.5/case9
CLASH Approach: **20/case5**, 17.25/case9, 14.5/case7, 7.5/case8, 7.5/case2

**NEW SITUATION 2** - [dangerous_driving(a),low_visibil,car(b),collision_ssr(b, a),cp(a, rear),cp(b, front),on_its_side(b)]
CAIN Approach: **0.78/case6**, 0.5/case5, 0.28/case3, 0.24/case7, 0.2/case9
CLASH Approach: **21.1/case6**, 12/case5, 8/case7, 6/case9, 2/case3

**NEW SITUATION 3** ---- [roundabout(a), lorry(a), lorry(b), enter_roundabout(b), cp(a, front), cp(b, front)]
CAIN Approach: **0.8/case9, 0.8/case8**, 0.49/case7, 0.4/case2, 0.34/case1
CLASH Approach: **24/case9**, 22.5/case8, 11/case7, 7.5/case2, 5/case5

New situation 1 represents an accident on a road junction (the fact 'from_the_right(V1,V2)' only occurs if the accident takes place on a road junction, roundabout or driveway) in which a car a moves from the right of a car b. The collision point on a is on its rear and on b on its front. Both cars are moving on their side of the road.
Applying Cain's metric the highest similarity value was for case 3[2]. This is an irrelevant case as in this accident a head-on collision occurred between a vehicle a and a car b that had to give way to vehicle a - an ambulance (this was not mentioned in the situation for case 3 but was considered in the outcome).

CLASH selected the best case in memory. Case 5 represents an accident in which a car a and a car b were moving on a straight road. Car a stopped and car b collided on the rear of car a. In this case car b was assigned the offence of not guarding a safe distance between it and the car ahead. Case 5 has the correct outcome for situation 1.

This interesting result is due to the relevance assigned by explanations to some facts in the past situation. CLASH performed better than Cain's metric due to the same reason that caused case C1 to be assigned a higher score than C2 by CLASH in fig. 3. As in C1 and C2 case 3 does not have complete or partial explanations whereas case 5 has a complete set of explanations.

[1] CLASH has proved not to be very sensitive to parameters $\lambda$ and $\mu$ provided they are much greater than $\kappa$.

[2] Due to space limitations we do not list some cases referenced in this example.

In the new situation 2 the driver of vehicle a is performing dangerous driving on a low visibility place and collides with a car b that is moving on the same side of the road. The collision points are, respectively, the rear and the front of a and b. When the accident occurs car b is on its side of the road.

For this accident case 6 had the highest score in both approaches and this was the correct selection.

The last new situation describes an accident on a roundabout between a lorry a that is going round on it and a lorry b that is entering the roundabout. A head-on collision occurs between them.

For this accident Cain's metric proposes two winners, cases 8 and 9. Case 8 is a case in which a car b was entering the roundabout and collided with a bicycle a that was going round on it. All streets that ended on the roundabout had a "junction ahead" sign. Due to this, car b was blamed of not respecting the sign. Case 9 is one in which a car b was entering a roundabout and a car a was going round on it. A head-on collision occurred between them. Blame was assigned to car a for not respecting the priority owed to a car that was traveling from the right of it.

Case 9 was the relevant case for the new situation outcome and CLASH assigned it the highest score. The reason for the discrimination made by CLASH between cases 8 and 9 was similar to the one made between cases A1 and A2 in fig. 3. Cases 8 and 9 had two unmatched facts in the matching situation, respectively, 'roundabout_wth_prior' (means that all streets that end on the roundabout have a "junction ahead" sign) and 'car(b)'. 'roundabout_wth_prior' belongs to a strong situation snippet in case 8 with cardinality 3 and 'car(b)' to a strong situation snippet in case 9 with cardinality 4. This implied that in CLASH the absence of 'roundabout_wth_prior' in the new situation was more penalizing for case 8 than the absence of 'car(b)' in the new situation for the similarity value of case 9. Due to this case 9 is the winner in CLASH.

## 5. Conclusions

Case-Based Reasoning is a well suited approach when a perfect theory on the domain is not available and a report of past cases exists. As it has been shown in this paper it is important to take into account the imperfections in past case descriptions and explanations.

We sustain that three kinds of explanations must be considered for case indexing: broken explanations, partial explanations, and complete explanations.

In our approach the concepts of matching situation, strong, weak and undetermined situation snippet are central to the matching process. The proposed similarity metric is also supported on these concepts.

Our metric cumulatively uses three measures for similarity assignment. Each measure relates to a kind of matching facts in the matching situation. We report three kinds of facts depending on their membership to undetermined, weak or strong situation snippets.

The relevance function assigns higher relevance to facts that belong to smaller situation snippets. This proved to be a sounding heuristic in many retrieval scenarios.

The contribution of each matching fact to the similarity value is increased by the number of times it occurs in the situation snippets. This is an intuitively interesting option.

The empirical results reported in this paper and the results obtained with CLASH prototype at work confirmed the expectations that we had on the described similarity metric. In fact CLASH never selected a past case less relevant than the one selected by the Cain's metric and most times it selected a more relevant one.

## References


1. Cain, Timothy, Pazzani, M. J. and Silverstein, Glenn, *Using Domain Knowledge to Influence Similarity Judgments*, in Proceedings of a Case-Based Reasoning Workshop, Morgan-Kaufmann, 1991.
2. Duda, R., Hart, P., *Pattern Classification and Scene Analysis*, New York: Wiley, 1973.
3. Hammond, K., *Case-Based Planning: An Integrated Theory of Planning, Learning and Memory*, Ph. D. Dissertation, Yale University, 1986
4. Kolodner, J., *Retrieval and Organizational Strategies in Conceptual Memory: A Computer Model*, Hillsdale, NJ.: Lawrence Erlbaum Associates, 1984.
5. J. Kolodner, and C. Riesbeck, *Experience, Memory, and Reasoning*, Lawrence Erlbaum Associates, Hillsdale, NJ, 1986.
6. Lebowitz, M., *Concept Learning in an Rich Input Domain: Generalization-Based Memory*, in Michalski, R., Carbonell, J., and Mitchell T. (Ed.), Machine Learning, Vol. 2, Los Altos, Ca.: Morgan Kaufmann Publishers, 1986.
7. Mitchell, T., Keller, R. and Kedar-Cabelli, S., *Explanation-Based Learning: A Unifying View*, Machine Learning, vol. 1 (1), 1986
8. Pazzani, M., *Creating a Memory of Causal Relationships: An Integration of Empirical and Explanation-Based Learning Methods*, Hillsdale, NJ.: Lawrence Erlbaum Associates, 1990.
9. Redmond, Michael, *Distributed Cases for Case-Based Reasoning; Facilitating Use of Multiple Cases*, in Proceedings of the National Conference on Artificial Intelligence, AAAI Press/The MIT Press, 1990.
10. C. Riesbeck, and R. Schank, *Inside Case-Based Reasoning*, Lawrence Erlbaum Associates, Hillsdale, NJ, 1989.

# STRUCTURAL SIMILARITY AS GUIDANCE IN CASE-BASED DESIGN *

Katy Börner
HTWK Leipzig, Dept. of Informatics
P.O.Box 66, 04251 Leipzig, FRG
katy@informatik.th-leipzig.de

**Abstract**

The effectiveness of case-based reasoning CBR depends on the ability to determine former experiences (cases) that are useful and applicable to solve new, similar problems. When one tries to handle *synthesis* tasks as opposed to *analysis* tasks, however, the determination of similarity alone is not enough: It becomes important to determine the *adaptability* of former cases to problems of current interest. Consequently, during similarity assessments rule-based knowledge concerning possible adaptations of previous cases becomes necessary. The objective of this paper is to present a new approach which interactively integrates and tunes case-based and rule-based knowledge in order to solve synthesis tasks. Structural similarity will provide guidance to solution adaptation. We will flesh out the general ideas of this approach and will show their motivation by pointing out relations to prior work. For illustrative purposes, we take an example of industrial building design.

## 1 Introduction

The purpose of this paper is twofold: The first is to introduce a close integration of case-based and rule-based background knowledge tuned to supplement each other[1]. The second is to present an approach to determine structural similarity to guide solution adaptation. Therefore, similarity is no longer defined as a value between 0 and 1 but as the most specific structure two cases have in common, inclusive of the modification rules needed to obtain this structure from the two cases.

The paper is organized as follows: First, we describe the main procedure that uses structural similarity as guidance to adapt prior solutions so that they fit new problems, and we point out relations to prior work. Second, we exemplify our approach, we solve a specific synthesis task taken from the domain of building design. Finally, we delineate a number of directions for future work.

## 2 Our Approach: Structural Similarity as Guidance

To introduce our general approach, we use Fig. 1. In the figure, the case-base is given on the left side. On the right side, the new problem including its solution is presented. We distinguish three different schemes of case representation: attribute-based, structural, and structurally modified. The more general these representations are, the more rounded the corresponding boxes are shown. Rules will be stored in a rule-base, as shown in the middle of Fig. 1. Arrows are used to mark the steps in case-based problem solving.

*This research was supported by the German Ministry for Research and Technology (BMFT) within the joint project FABEL under contract no. 413-4001-01IW104. Project partners in FABEL are German National Research Center of Computer Science (GMD), Sankt Augustin, BSR Consulting GmbH, München, Technical University of Dresden, HTWK Leipzig, University of Freiburg, and University of Karlsruhe.

[1]Note, that by tuning both kinds of knowledge their representation and use will be different from stand-alone case-based or rule-based problem solvers.

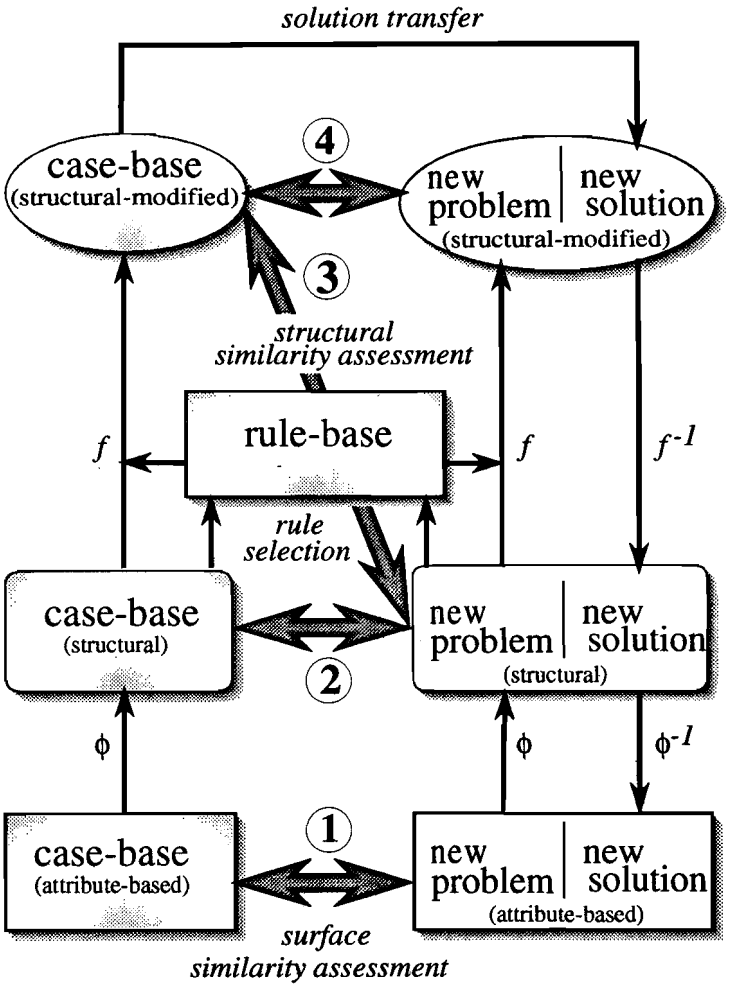



Figure 1: General approach and different levels to assess similarity

As a basis for reasoning cases and rules are needed. Cases have to be represented both in attribute-based and structural forms (e.g., by terms, trees, graphs). Background knowledge is represented by domain-dependent and task-dependent rules (e.g., term, tree, or graph substitutions and generalizations) including their 'inverse' rules.

The main procedure to assess structural similarity and adaptation is as follows: Given the new problem in attribute-based description, we start by determining a set of candidate cases. The surface attributes of those cases are similar to those of the new problem. Based on this computationally cheap analysis (*surface similarity assessment*) of the problem, we can now use transformation function $\phi$ to translate the new problem into a structural representation. Corresponding to the new problem and the preselected candidate cases, modification rules will be chosen (*rule selection*) and applied until a common structure of the actual problem and one candidate case is found. Now the solution of this candidate case, likewise modified, can be transferred to the new problem. After that, inverse modification rules $f^{-1}$ are applied to get the concrete structural representation of the new solution. Using $\phi^{-1}$ we will get the attribute-based representation of the new solution representation. This will be offered to the user.

The core idea of our approach does not refer to the way similarity is assessed but to the way similarity will be used to lead to adapted, structurally sound solutions. The common structure of cases together with the modification rules applied to obtain them determine which prior solutions are useful. The inverse modification rules will show how to adapt them.

Also sketched in Fig. 1 are prior approaches to determining similarity (denoted by fat grey double arrows): (1) In CBR, *surface similarity assessments* based on attribute-based representations are frequently used (c.f., [11, 10, 7]). (2) If interdependencies of attributes have to be taken into account, representations like terms, trees, and graphs are used as basis for similarity assessments. (3) There are approaches where cases stored in the case base are modified (e.g., using unification [6] or using letter substitution rules as known in speech recognition) to determine similarity. (4) The principle of redescription [5, 9] modifies new and old problem descriptions depending on the problem at hand. We emphasize that all the approaches mentioned above deal with analysis tasks, and therefore synthesis and adaptation issues are not addressed.

# 3 An Example: Case-based Industrial Building Design

Much work has been done in case-based building design [3, 2, 8], which is one of the most complex real world synthesis tasks. In our project, we focus on the installation of supply system nets in industrial buildings with a complex infrastructure. The main problem is how to layout subsystems for fresh and return air, electrical circuits, warm, cold, and used water, etc.. By using the A4 model introduced in [4] and letting thinly drawn circles denote places where accesses can be placed, and letting ellipses denote areas where connections of supply accesses can be placed, the task of designing arrangement of connections for supplies that cover all of a given set of accesses for supply is reduced to the connection of circles with ellipses.

To tackle this task, we use two different types of case representations as well as a rule-based representation of the background knowledge. The first of the case representations is an *attribute-based* representation of visually prominent features of objects. Following the work of LUDGER HOVESTADT [4] each object (circle or ellipse) will be represented by its spatial dimensions and nine further attributes like *time* at which this object was created, *aspect* which assumes one of 'return air,' 'fresh air,' etc., and *morphology* which refers to 'access,' 'connection,'.., etc. This fixed set of dimensions will be used as indices.[2] This representation will be used to produce graphics, the main basis for man-machine interaction in building design.

Second, we have to encode *structural* knowledge, e.g., case-based knowledge about spatial arrangements and relative positions of objects in a machine-usable form. Our approach, which is influenced by the work of BIPIN INDURKHYA (cf.[5]), is to represent the complex structures like supply air net structures as terms over some appropriately tailored signature. A finite, heterogeneous, and finitary signature is assumed. This is taken as a basis for building terms and formulae, as usual[3]. Additionally, equational knowledge about functions and their relations is formalized to represent term rewriting knowledge. Note that a solution description contains the corresponding problem description. There is a function $\phi$ with its inverse which realizes the transformation of the attribute-based descriptions into structural ones and opposite.

Third, we need background knowledge *rules* for determining proper domain dependent and task dependent modifications of structural case representations. Terms can be modified using generalizations. To express generalized terms we need a sorted family of variables. For simplicity, we assume all variables to be called $x$, with indices whenever necessary. There are meaningful adaptations like *reflection, rotation, translation*, etc. in our domain. Additionally, structural representations can be modified using abstraction rules, which transform term expressions to constants (abstract attributes) like *row, regular, covered* etc. These three different kinds of modification rules including their inverses will be stored in the rule-base.

Given these three types of knowledge representations, we are able to determine structural similarity and use it to guide the solution adaptation. For illustration, the main procedure given in Fig. 1 is exemplified in Fig. 2. The left, lowest box shows the pictorial and attribute-based representation[4] of one typical case stored in the case-base. By taking the functions *cover* and

[2] To get cases in a less redundant form, 'space-coordinates' will be normalized. Therefore, we simply assume that the smallest x-, y-, and z-dimensions of each case is equal to zero.

[3] The detailed formal description of the signature used to represent cases structurally can be found in [1]

[4] For simplicity, we only gave the values of the atributes x, dx, y, dy, time, aspect, and morphology of each

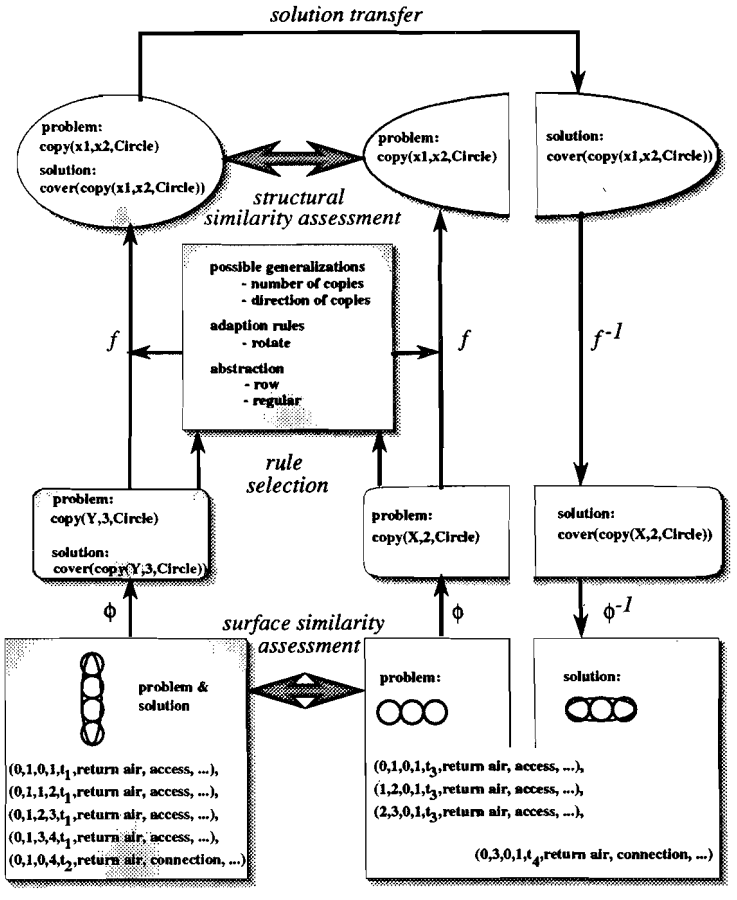



Figure 2: Structural similarity assessment and adaptation - an example

*copy* and object constants and combining them with appropriate parentheses and commas, we are able to express the solution of this case structurally by *cover(copy(Y,3,Circle))*. This term stands for *take one circle, copy it three times and arrange all in y-direction. Afterwards cover all circles with a single ellipse.* The right, lowest box shows the pictorial and attribute-based representation of the new problem to be solved. Given in the same box but not available at this time is the solution of the new problem. The particular intention (also called subgoal) the user wants to concentrate on is the connection of air supplies. The first initial analysis of the new problem can be done on the basis of the inexpensive *surface similarity assessment*, based on the attribute-based descriptions. The result is a set of candidate-cases which have similar surface attributes such as aspect, number of objects etc. In this way, the rather large set of cases stored in the case-base can be reduced to a few useful candidate-cases. The next step is the transformation of the new problem into a structural representation. Here, candidate-cases provide information about proper transformations referred to as $\phi$. Thus, the new problem, which consists of three circles arranged in a row in x-direction can be structurally represented by *copy(X,3,Circle)*.

Based on the structural representation of superficially similar prior cases, the more expensive **structural similarity assessment** is performed. Axioms and modification rules will be applied to determine the main structure the new and a prior problem have in common. In our example, there are at least three different ways to achieve this:

- The first way uses generalization. For example, the concrete arrange direction and

object.

the number of copies will be replaced by variables $x_1$ and $x_2$. The resulting common problem description will be: *copy($x_1$,$x_2$,Circle)* as shown in Fig. 2.

- The second way is to use generalization and adaptation rules. Here, the number of copies will be generalized, too. One term representation will be rotated about 90 degrees.
- The third way uses abstraction. Here abstract descriptions like *row* and *regular* will be used as structural attributes. The idea behind this is that the more identical abstract attributes structural descriptions share, the more similar they are.

Given the main structure of both problem descriptions, we can simply transfer the main prior structurally modified solution (in the example *cover(copy($x_1$, $x_2$,Circle))*) to the actual problem (in Fig. 2 referred to as *solution transfer*). Using the knowledge about the sequence of modifications to determine the common structure, the transferred solution can be adapted to the new problem. This is denoted by $f^{-1}$. To get the concrete structural solution, in the example,

- where two generalizations were used to determine structural similarity, one has to replace $x_1$ by its former value $X$ and analogous $x_2$ by 2. The resulting term will be *cover(copy(X,2,Circle))*.
- where generalization and adaptation were used, one applies the inverse adaptation function and rotates the figure about -90 degrees (or 270 degrees) and replaces the variable number of copies by *2*. Likewise, the resulting term will be *cover(copy(X,2,Circle))*.
- where abstraction was used, the transferred solution can be expressed by the attribute *covered*. But the reverse concretization is somewhat difficult. Given terms and their corresponding abstract descriptions, one can try to find one term-representation which fulfills all abstract attributes (in this example *row, regular*, and *covered*). This suffices, if the number of these term-attribute assignments remains small but becomes intractable otherwise.

Given the structural representation of the new solution the application of the inverse transformation $\phi^{-1}$ yields the attribute-based and hence pictorial representation of the new solution.

# 4 Conclusion and Future Work

The approach introduced in this paper offers a practical way to integrate and tune case-based and rule-based background knowledge to solve real world synthesis tasks. Resulting advantages are problem solutions in synthesis domains like design, where only locally consistent knowledge is available. Even with only locally consistent knowledge, the adapted solutions are not necessarily bad solutions, because just the appropriate rules can be selectively applied to adapt them. In addition, the structural similarity assessment provides a basis for more descriptive explanations for why particular solutions have been adapted.

There are some interesting directions of further work. Some knowledge structures in our domain cannot be efficiently captured by term representations. Therefore, we wish to extend our approach to other knowledge representation schemes like general trees, graphs, etc. In such cases, different cases will have completely different structural representations and hence different modification rules. However, the main direction for further work is the integration of learning. The domain specific determination of knowledge representations and interactions between them is a first step before learning can be included.

To demonstrate the effectiveness of our approach, we have started implementing a system called *SynTerm* (like *Syn*thesis by using *Term* representations). This program realizes problem solving in industrial building design using the approach introduced.

# Acknowledgements

This research is part of and has been strongly inspired by work done in the project FABEL the general objective of which is the integration of case-based and model-based approaches in

knowledge-based systems. This work was guided by BIPIN INDURKHYA's work on analogical reasoning. He contributed to this work by providing encouraging feedbacks on an earlier version of this paper. I wish to thank DIETMAR JANETZKO for his helpful comments on an earlier draft. I am indebted to my advisor KLAUS P. JANTKE for his continuing support and encouraging feedback. Additionally, I would like to thank NAOKI ABE for his help in improving the presentation of this paper.

## References

[1] Katy Börner. Structural similarity in case-based building design. Submitted to the German Workshop on CBR. 1993.

[2] Eric A. Domeshek and Janet L. Kolodner. A case-based design aid for architecture. In *Proc. Second International Conference on Artificial Intelligence in Design*, pages 497–516. Kluwer Academic Publishers, June 1992.

[3] Ashok K. Goel. *Integration of case-based reasoning and model-based reasoning for adaptive design problem solving.* PhD thesis, Ohio State University, 1989.

[4] Ludger Hovestadt. Armilla4 - an integrated building model based on visualisation. In *Proc. EuropIA '93, Delft, The Netherlands*, page to appear, 1993.

[5] Bipin Indurkhya. On the role of interpretive analogy in learning. *New Generation Computing*, 8(4):385–402, 1991.

[6] Klaus P. Jantke. Nonstandard concepts of similarity in case-based reasoning. In *17th Annual Meeting of the German Society for Classification in Kaiserslautern*, page to appear, March 1993.

[7] Janet L. Kolodner. An introduction to case-based reasoning. *Artificial Intelligence Review*, 6:3–34, 1992.

[8] D. Navin chandra. Innovative design systems, where are we and where do we go from here? Part I: Design by association. *The Knowledge Engineering Review*, 7:3:183–213, 1992.

[9] Scott O'Hara. A model of the 'redescription' process in the context of geometric proportional analogy problems. In K. P. Jantke, editor, *Proceedings of the International Workshop on Analogical and Inductive Inference*, pages 268–293. Springer-Verlag, 1992.

[10] Michael M. Richter. Classification and learning of similarity measures. In *Proc. der 16. Jahrestagung der Gesellschaft für Klassifikation e.V.* Springer Verlag, 1992.

[11] Stefan Weß. PATDEX/2: Ein System zum adaptiven, fallfokussierenden Lernen in technischen Diagnosesituationen. SEKI-Working Paper SWP91/01, Dept. of Computer Science, University of Kaiserslautern, Germany, January 1991.

# An Under-Lying Memory Model to Support Case Retrieval.

Mike G. Brown[1]

Department of Computer Science, University of Manchester
email michaelb@cs.man.ac.uk

**Abstract.** The work described in this paper is aimed at providing an underlying model of memory to support Case-Based Reasoning (CBR). The approach taken is to include a number of types of biasing constraint within the structure of memory itself and to use an activation passing process to exploit this information for retrieval of relevant cases. This provides the potential for highly flexible case retrieval without resorting to exhaustive search of memory. This claim is supported by initial experimentation using a prototype implementation of the model of memory.

## 1 Introduction.

CBR is now a well established problem solving technique. Part of its popularity lies in the intuitive appeal of a computer system that can base its reasoning on the reuse of isolated 'experiences'. Yet this intuitive appeal may also be deceptive. Each of the stages of the CBR process (such as retrieval, relevancy judgement, evaluation, adaptation and learning) is in itself a complex task that has spawned and continues to support many research projects.

The work described in this paper seeks to find a more fundamental mechanism that underpins the process of CBR. Research along these lines may help to clarify the interaction between the different stages of CBR and also determine the relationship of CBR to similar types of reasoning, such as analogical reasoning.

It is desirable that an underlying model of memory has the following properties:

- **Flexibility.** There should be a minimal inherent restriction in terms of the circumstance under which a case can be retrieved and hence reused
- **Generality.** The knowledge representation should not restrict what constitutes a case.
- **Efficiency.** The retrieval of appropriate cases should avoid an exhaustive search of memory.

In order to satisfy these potentially conflicting goals the approach taken in the described work is to use a variety of constraints as *biases* on retrieval. This approach is similar to that taken in systems such as PARADYME [8, 7] and ARCS [14]. However, the work described in this paper is novel in so much as it attempts to satisfy the above requirements by the exploitation of a richly structured memory and as such the proposed model of retrieval is potentially less computationally intensive.

The model of memory that will be described in this paper is illustrated in figure 1. The retrieval mechanism has three distinct phases. Phase 1 simply involves the access of all components of the target case's decompositional structure[2]. The second and most prolonged phase is a search through the network of memory from each of the target case component to 'similar' source case components. The types of information that influence this process and hence contribute to a biasing towards similarity are described in section 2, while the activation passing mechanism that performs retrieval is described in section 3. The final phase is for all the isolated retrievals of source case components to be combined to generate a global measure of the retrieval strength for each source case. This process is briefly described in section 3.2. The model of memory has been implemented in the CRASH[3] prototype system and some initial experimental work is described in section 4.

[1] This work was supported by the Scientific and Engineering Research Council (SERC).

[2] Note that it is assumed that the assimilation of the target case with the rest of memory has already occurred.

[3] Case Retrieval by Activation passing SHell.

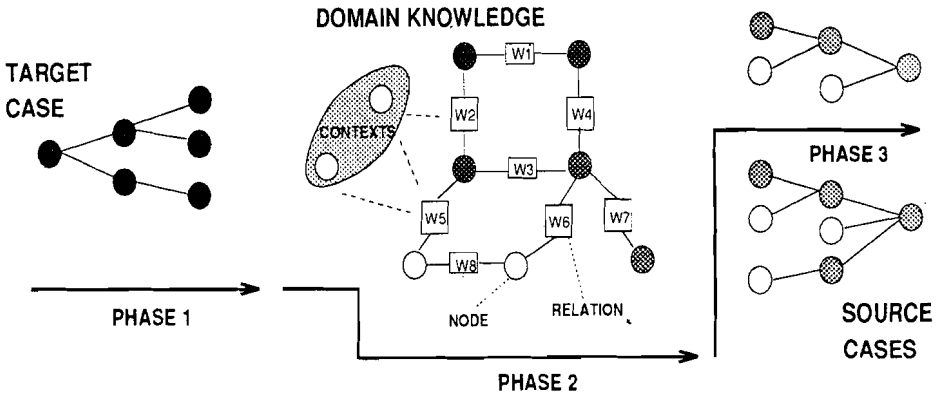



Figure 1: The Memory Structure and Retrieval Flow for CRASH.

## 2 Memory Structure and Retrieval Biasing.

Five distinct classes of information are recognised as inherent to the proposed model of memory. Each of these information categories is described below in terms of how it is represented and the role it plays in case retrieval.

- **Semantic Knowledge.** This is the "factual" knowledge held in memory, including the cases themselves and all appropriate supporting domain knowledge. For CRASH, this knowledge is represented as a conceptual graph. A 'local' representation is used where each node represents exactly one item of knowledge and where the relations in the network are analogous to the associative relationships that exist between knowledge items in reality. In this way the basic structure of the conceptual graph defines a search space for retrieval of knowledge that is meaningful.
- **Contextual Delimitation.** In reality few, if any, "facts" are universally true, rather a given item of knowledge is only appropriate *in context* In the CRASH model of memory *"context"* equates to the explicit representation of knowledge that determines when an individual relationship is relevant. It follows that the general role that context plays is to *segregate* memory so that appropriate knowledge can be focussed on at the time of retrieval. The fact that the constraining action of context can be dynamically superimposed upon memory means that there is greater flexibility in terms of the use of memory than can generally be achieved using traditional *"indexing"* approaches.
- **Recently Retrieved Knowledge.** The knowledge in memory that has already been selected plays an important role in determining what new facts should also be retrieved [9]. Hence this type of information acts as a fluid form of context for the on-going retrieval process. For CRASH, this type of information equates to the momentary distribution of activation during retrieval.
- **Descriptive Structure.** It is well recognised, particularly in analogical reasoning [5], that the two cases are more likely to strongly relate to one another if their descriptions involve similar patterns of relations. It follows that retrieval should attempt to preserve structure of a target case when retrieving applicable source cases. The achievement of this in CRASH is discussed in section 3.2.
- **Typicality and Idiosyncrasy.** These are two meta-level measures based on *all* knowledge held in memory. These factors provide a default, graded structure for memory that, in the absence of any stronger contextual constraints, can be used to guide retrieval along the paths through memory that are potentially of most use. Typicality is useful in terms of providing guidance of retrieval towards most frequently encountered semantic knowledge. This can be shown to improve the accuracy for abductive inference of additional knowledge about a target case. By contrast, idiosyncrasy guides retrieval to the most exceptional (and hence characteristic) features of a particular description. Both measures are assigned to all relations in the conceptual graph by statistical calculations in CRASH. Details of the calculation and use of these relation weights can be found in [2, chapter3].

## 3 On the Use of Activation Passing for Retrieval.

There are a number of reasons why activation passing was chosen as a suitable technique for implementing retrieval. The numeric markers that are passed provide a suitable vehicle by which the various constraints

built into the organisation of memory can be quantitatively assessed *during* retrieval. This has important efficiency and flexibility implications with respect to retrieval because, if memory is suitably well structured, the need for post-retrieval evaluation of cases may be avoided. In this way CRASH improves upon systems such as PARADYME [8, 7] and MAC/FAC [6].

The responsiveness of activation passing to constraints can greatly be increased through the introduction of competition into the process [10]. However, traditional approaches to competitive activation passing are not well suited to the task of case retrieval and an appropriate solution is put forward in section 3.1. An additional problem with activation passing is that it is difficult to include a bias towards preserving descriptive structure during retrieval. A possible solution to this problem is described in section 3.2.

### 3.1 Competitive Activation Passing for Case Retrieval.

A major problem in applying activation passing techniques to the task of case retrieval is the determination of an appropriate form of 'competitiveness'. In order to be space efficient some form of *"virtual"* inhibition is required [12], rather than the reliance on explicit inhibitory links such as in [14, 10]. However, the virtual inhibition described in [12] produces a "winner-takes-all" strategy that is too restrictive for case retrieval: for successful CBR it is often best to deal with a small corpus of potentially relevant cases rather than the one most relevant case [11, 3, 13].

The adopted solution is to systematically restrict the number of outward links that can be used by a node at any point during phase 2 of the retrieval. There are several factors that determine which of a given node's outputs will be selected as receivers when it sends activation. First and foremost an output is only a candidate receiver if the currently selected contexts explicitly state that the relation associated with that output is relevant[4]. Secondly, any candidate receiver output that leads to a node that already possesses activation will automatically be selected. This instigates the desired bias in retrieval towards the reinforcement of already retrieved knowledge. In particular it favours the recollection of coherent bodies of facts, rather than facts in isolation. Finally, if the above selection criteria does not exhaust the allowed limit on a given node's receivers, outward links are selected in the order imposed by the typicality and idiosyncrasy relation weighting, up to the point at which the receiver limit is reached.

Once all receivers have been selected the same amount of activation is sent to each. The competition therefore lies in becoming a receiver, not in terms of acquiring more activation than other receivers. In particular, relation weighting has the role of controlling in what direction an activation distribution expands but has no effect on the amount of activation that is sent. This means that, in a suitably unusual context, a low weighted relation may be one of the few selected receivers and subsequently a relatively large amount of activation will pass through it. In this way the flexibility of the proposed retrieval mechanism is enhanced.

### 3.2 Preserving Case Structure During Activation Passing.

A second major problem with activation passing is that, because it is governed by highly localised rules, it is difficult to match source and target cases based on their descriptive structure. A bias towards this type of structural preservation is implemented in CRASH through the labelling of disjoint activation distributions by *"colour"* tags.

A dependency hierarchy between different activation colours is generated during phase 1 of the retrieval reflecting the decompositional structure of the target case itself. The propagation of activation from each target case component to like source case components then proceeds during phase 2 of the retrieval, more-or-less independently. However, in phase 3 the isolated activations deposited in the nodes representing components of source cases are amalgamated by the propagation of activation up the source case structures. The colour dependencies generated in phase 1 can be used to guide the recombination of colours during phase 3. This can be used to ensure that only source cases that are highly isomorphic to the target case in terms of their representation can be retrieved. More generally, the strength of merging of two colours depends on their proximity in a colour dependency hierarchy. This provides some tolerance to deviation between source and target case structures.

## 4 Experimental Work.

This section gives a brief description of some of the experimental work that has so far been carried out using the CRASH prototype implementation of the model of memory.

[4] The selection of contexts is beyond the scope of this paper, see [2] for details.

### 4.1 Experiment Description.

For the purposes of initial experimentation a simple *"bin-packing"* problem was devised. This problem involves a rectangular grid and a set of right-angled polygons. The task is to place all the shapes within the grid so that they tessellate and exactly cover the grid. The problem is simplified such that the polygons are either rectangles or L-shapes of various sizes.

For the problem to be tackled using CBR requires a collection of source cases that represent individual placements of shapes within a grid. Accordingly a set of source cases were produced by recording the problem-solving actions of the author while tackling a number of exemplar bin-packing problems set by a colleague. Each source case is composed of two main parts, an initial state and the placement. The initial state in turn comprises a description of the remaining space to be filled and a description of the set of shapes still to be placed. The placement is composed of a shape selected for placement and the shape's final position within the remaining grid. The representation of cases was carefully designed so as to minimise the amount of bias towards problem solving characteristics that are implicit within the case description structure itself.

A target case was encoded and a series of retrievals performed. Experimentation was performed across a range of different cases bases and with systematic variation in the key parameters that control the operation of CRASH. In particular, the amount of retrieval effort was varied between retrievals by altering the allowable number of receivers per node. The result of each retrieval is a graded ordering of the source cases in terms of how much activation resulted within their representation.

### 4.2 Experimental Bench-marks.

This ordering of source cases produced by each retrieval is compared to three theoretical bench-mark orderings defined in terms of the following criteria:

- **Semantic Ordering:** This is based on a measure of the 'semantic similarity' between target case and each source case. Cases are ordered according to a measurement of the maximal size for a set of compatible feature pairings that can be generated between two case descriptions.
- **Structural Ordering:** This is based on a measure of the correspondence between the structure of the target case description and each source case description.
- **Pragmatic Ordering:** During knowledge elicitation for the bin-packing problem a set of heuristics were identified that can be used to achieve a high success rate when generating solutions. Examples of these heuristics include *"position the largest remaining shape next"* and *"keep the remaining space as rectangular as possible"*. These heuristics can be used to evaluate how *useful* is the placement suggested by each source case with respect to the initial state of the target case which in turn provides a pragmatic ordering of source cases.

The deviation of the retrieval ordering from each bench-mark ordering is calculated. This measure can be converted into a biasing strength by considering the probability that a randomly generated ordering of the source cases has as close a match to the bench-mark ordering as that produced by a given retrieval.

### 4.3 Results and Analysis.

The results for all retrievals so far carried out using CRASH on the bin-packing problem are summarised in figure 2. One of the main goals of the experimentation is to empirically establish a relationship between the amount of retrieval effort performed (with respect to an exhaustive search of memory) and the quality of retrieval. The results shown in figure 3 are typical[5].

Several conclusions can be made from these results. Firstly it is clear from figure 2 that there is generally a strong positive bias towards all three bench-marks. In particular, as is shown in figure 3, the correspondence between structural and semantic ordering is high. This coupling can be attributed to the use of a standard format for representing bin-packing cases.

The case retrieval also shows a general bias towards selecting the source cases that are of most use for problem solving. This helps confirm an underlying premise of this work: that the combination of various preference constraints can be sufficient to determine what source cases are *relevant* to a new problem solving episode. Figure 3 shows that the bias of retrieval towards pragmatic ordering is generally slightly

[5]Figure 3 shows the results for a series of 39 separate retrievals carried out on a case base containing 22 bin-packing cases.

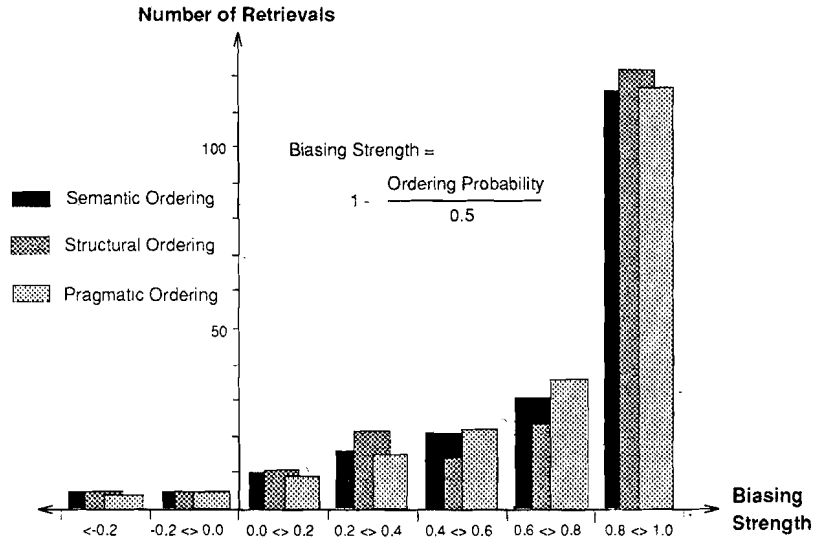



Figure 2: The Overall Quality of Retrieval for the Bin-Packing Case Study.

weaker than for the other two bench-marks. This is not surprising and reflects the efforts to avoid any undue pragmatic bias in the structure of the case representation. In addition, the context sequence used to control phase 2 of all experimental retrievals implements a feature-based search from *all* components of the target case. To hone the retrieval for problem-solving would require retrieval strategies that prioritized the target case components; for example, retrieving source cases from the larger target case shapes but ignoring the smaller shapes.

Finally, and perhaps most importantly, figure 3 shows that a near maximal retrieval quality can be produced with respect to all bench-marks at a retrieval effort well below that required for exhaustive search of memory ( $\approx 7\%$ in the example of figure 3). This is strong evidence for the claim put forward at the start of this paper: that a flexible model of retrieval can be implemented by relying on a rich organisation of memory rather than on a more brute-force approach involving an extensive search of memory.

# 5 Conclusions and Future Work.

This paper describes the work that has been carried out in the development of an underlying model of memory to support CBR. The potential for utilising activation passing for the achievement of retrieval has been recognised elsewhere [14, 4, 1, 15]. However the scheme proposed here is unique in that it does not rely on the explicit representation of inhibitory links yet is sufficiently selective to yield useful case retrieval without resorting to exhaustive search.

The initial experimental results support the claims that the proposed model can efficiently incorporate into retrieval various measures relating to similarity judgements (and hence relevancy). However there is a need to apply the model to larger and more complex problem domains. This is intended as one area for future work.

From a theoretical stand-point an important future direction is to investigate how other aspects of CBR unify with retrieval. For example, the tasks of retrieval and mapping seem intuitively to be linked. Furthermore, it may be plausible that tasks such as case adaptation can be recast as iterative retrieval, provided that the relevant knowledge is held in memory. The approach of seeking to explain the process of CBR via such a unified model may help to generate important insights into the general role of memory in reasoning.

# References

[1] Lee Becker and Kamaran Jazayeri. A Connectionist Approach To Case-Based Reasoning. In *Proceedings of the Case-Based Reasoning Workshop*, pages pp213–217, Pensacola Beach, Florida, 1989. Morgan Kaufmann Publishers.

[2] Mike G Brown. *A Memory Model for Case Retrieval by Activation Passing*. PhD thesis, University of Manchester, 1993. Submitted for examination in October.

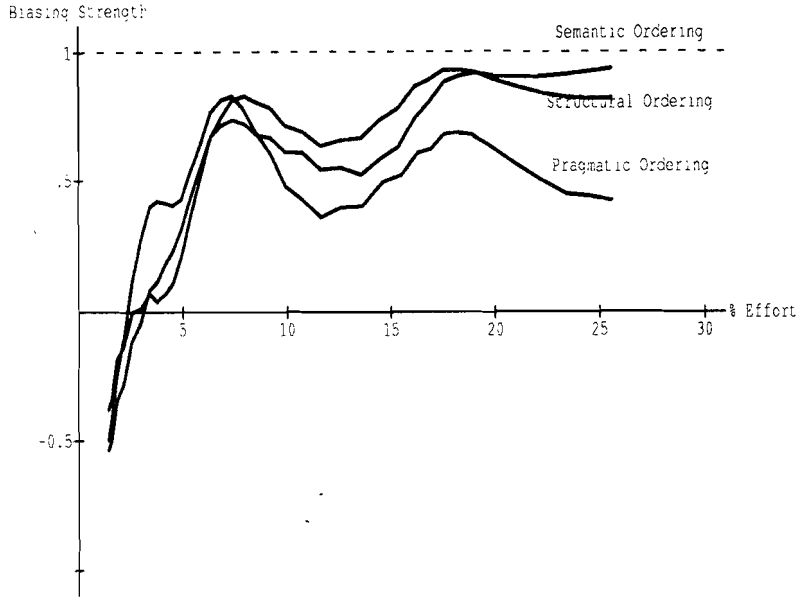



Figure 3: A Plot of Retrieval Effort vs Retrieval Quality.

[3] Diane J Cook. The Base Selection Task In Analogical Planning. In *IJCAI 91, Proceedings of the 12th International Joint Conference On Artificial Intelligence*, volume 2, pages pp790–795, Sydney, Australia, 1991. Morgan Kaufmann Publishers, Inc.

[4] Eric Domeshek. Parallelism for Index Generation and Reminding. In *Proceedings of the Case-Based Reasoning Workshop*, pages pp244–247. Pensacola Beach, Florida. 1989. Morgan Kaufmann Publishers.

[5] B Falkenhainer, K D Forbus, and D Gentner. The Structure Mapping Engine: Algorithm and Examples. *Artificial Intelligence*, 41(1):pp1–63, November 1989.

[6] Kenneth D Forbus and Dedre Gentner. MAC/FAC: A Model of Similarity-based Retrieval. In *Proceedings of the Thiteenth Annual Conference of the Cognitive Science Society.*, pages pp 504–509, Chicargo, Illinois, August 1991. Lawrence Erlbaum Associtates.

[7] Janet L Kolodner. Judging Which is the Best Case for a Case-Based Reasoner. In *Proceedings of the Case-Based Reasoning Workshop*, pages pp77–81. Morgan Kaufmann Publishers, 1989.

[8] Janet L Kolodner. Selecting the Best Case for a Case Based Reasoner. In *Proceedings of the Eleventh Annual Conference of the Cognitive Science Society.*, pages pp155–162, Ann Arbor, Michigan, August 1989. Lawrence Erlbaum Associates.

[9] Charles Eugene Martin. *Direct Memory Access Parsing*. PhD thesis, Yale University, 1991.

[10] James L McClelland and David E Rumelhart. An Interactive Activation Model of Context Effects in Letter Perception: Part 1 An Account of Basic Findings. *Psychological Review*, 88:pp375–407, September 1981.

[11] Michael Redmond.

[12] James A Reggia. Virtual Lateral Inhibition in Parallel Activation Models of Associative Memory. In *IJCAI-85. Proceedings of the 9th International Joint Conference on Artificial Intelligence.*, volume 1, pages p244–248, Los Angeles, California, August 1985.

[13] Katia P Sycara. Resolving Goal Conflicts via Negotiation. In *AAAI-88. Proceedings of the Seventh National Conference on Artificial Intelligence*, volume 1, pages pp245–250, Saint Paul, Minnesota, August 1988. Morgan Kaufmann Publishers, Inc.

[14] Paul Thagard, Keith J Holyoak, Greg Nelson, and Gochfeld. Analog Retrieval by Constraint Satisfaction. *Artificial Intelligence*, 46:pp259–310, 1990.

[15] Philip Thrift. A Neural Network Model For Case-Based Reasoning. In *Proceedings of the Case-Based Reasoning Workshop*, pages pp334 –336. Pensacola Beach, Florida. 1989. Morgan Kaufmann Publishers.

# Similarity Measures for Structured Representations

H. Bunke and B.T.Messmer
Institut für Informatik und angewandte Mathematik, Universität Bern,
Länggassstr. 51, CH-3012 Bern, Switzerland
bunke@iam.unibe.ch, messmer@iam.unibe.ch

## 1 Introduction

The main idea in case-based reasoning is to use the solution of a problem that has been solved earlier in order to solve a new problem. Given the actual problem $P$ and a collection of previously solved problems $P_1, P_2, \ldots, P_n$, one first evaluates the similarity between $P$ and each $P_i, i=1,\ldots,n$. Once the case $P_i$ has been found that is most similar to $P$, its solution is used in order to construct a solution of $P$. The similarity measures used in many case-based reasoning systems assume that cases are represented by collections of attribute-value pairs. Based on this assumption, the similarity between two cases is usually computed by a weighted sum of the similarity of the individual attribute values. For a general discussion of this type of similarity measures see [1].

In this paper, we propose a different approach. We assume cases not being given just by collections of attribute-value pairs but by structured representations. Formally, we assume each case being represented by a *directed labeled graph* (or graph, for short) $g = (N, E, \alpha, \beta)$, where

- $N$ is the finite set of nodes,
- $E \subseteq N \times N$ is the finite set of edges,
- $\alpha : N \mapsto L_N$ is the node labeling function,
- $\beta : E \mapsto L_E$ is the edge labeling function;

$L_N$ and $L_E$ are the finite alphabets of node and egde labels, respectively. Using such a representation, we normally represent concepts or objects of the problem domain by nodes, and relations between concepts or objects by edges. Relations can represent, for example, spatial, temporal, or causal relationships between nodes. The alphabets of node and edge labels are problem dependent and vary, in general, from one application to the other. Particularly, the above definition includes semantic networks and frame systems as special cases of graphs if we introduce relations like "instance","instance of", "a kind of" a.s.o.

## 2 A Similarity Measure on Graphs

Using graphs as introduced in the last section for the representation of problems, or cases, a measure is needed that gives the similarity of any two graphs. In this paper, we introduce a similarity measure based on a *weighted graph edit distance*. Our proposed measure is a generalization of string edit distance [2].

We start from a set of elementary edit operations on graphs, namely, the insertion, deletion, and substitution of a node or an edge in a graph. Formally, this set is given by

$$EO = \{del\_node, ins\_node, subst\_node, del\_edge, ins\_edge, subst\_edge\}.$$

If we apply one or more of these edit operations to a given graph $g_1$, a new graph $g_2$ is obtained. For example, the graph shown in Fig 1a can be transformed into the graph shown in Fig 1b by (1) substituting the node labeled "Cup" by a node labeled "Bowl", (2) substituting the edge labeled "below-of" by an edge labeled "right-of", (3) inserting a node labeled "Noodles", and (4) inserting an edge labeled "contains" between the nodes "Plate" and "Noodles". Apparently, our set of edit operations is complete in the sense that it allows to transform any given graph $g_1$ into any other graph $g_2$. This can be readily concluded from the fact that $EO$ contains the insertion and deletion of both nodes and edges. Thus, in order to transform a given graph $g_1$ into any other graph $g_2$, we could first delete all nodes and edges in $g_1$ and then insert all nodes and edges in $g_2$, for example. Note that for any two graphs $g_1$ and $g_2$, there are usually more than one sequence of edit operations transforming $g_1$ into $g_2$.

In order to model the fact that certain differences between two graphs have more weight or importance than others, we introduce *costs* for the basic edit operations. Let $e \in EO$ be an edit operation. Then $c(e) \geq 0$ denotes its cost. If it is required, one can normalize the costs such that $0 \leq c(e) \leq 1$. Given a sequence $s = (e_1, e_2, \ldots, e_n)$ of edit operations with $e_i \in EO, i = 1, \ldots, n$, we define its cost $c(s) =$

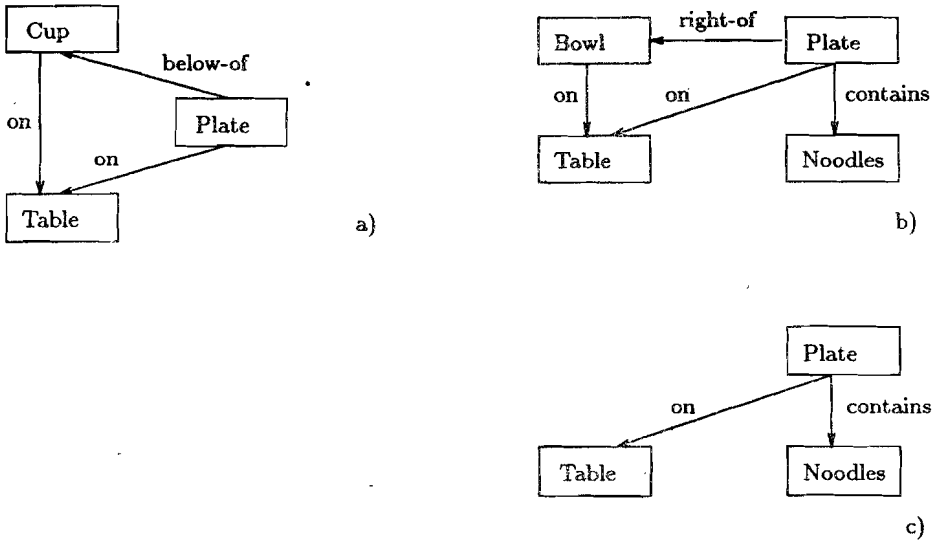



Figure 1: An example illustration of basic edit operations and subgraph isomorphism.

$\sum_{i=1}^{n} c(e_i)$. Finally, for any two graphs $g_1$ and $g_2$, we define their *edit distance* $d(g_1, g_2)$ as the minimum cost sequence of edit operations that transform $g_1$ into $g_2$. Formally,

$$d(g_1, g_2) = min\{c(s) | s = (e_1, \ldots, e_n) \text{ is a sequence of edit operations transforming } g_1 \text{ into } g_2\}.$$

Intuitively, the cheaper and the fewer the operations are that are required to make $g_1$ and $g_2$ identical, the smaller is the edit distance $d(g_1, g_2)$ between $g_1$ and $g_2$.

The graph distance defined above has a number of interesting properties, like reflexivity, symmetry, or metric property, depending on the way the costs of the operations in $EO$ are defined. If we define the cost of any identical substitution equal to zero and the cost of any other operation greater than zero, then $d(g_1, g_2) = 0$ if and only if $g_1$ and $g_2$ are isomorphic to each other. Similarly, if not only the costs of identical substitutions but also the costs of any insertions in $g_2$ are equal to zero, while all other operations from $EO$ have costs greater than zero, then $d(g_1, g_2) = 0$ if and only if $g_1$ is a subgraph of $g_2$.

Apparently, $d(g_1, g_2)$ is rather a measure of dissimilarity than similarity between $g_1$ and $g_2$. However, it can be easily converted into a similarity measure $s(g_1, g_2)$ by defining, for example, $s(g_1, g_2) = [d(g_1, g_2)]^{-1}$. It is also easy to normalize $d(g_1, g_2)$ or $s(g_1, g_2)$ such that all values are restricted to a certain interval, for example, the interval $[0, 1]$.

## 3 A Practical Procedure for Subgraph Isomorphism Detection

The concept of graph distance introduced in the last section is very flexible and powerful. However, its actual computation is not trivial. A possible approach to graph edit distance computation is graph search[1]. When computing $d(g_1, g_2)$ by means of graph search, we systematically explore all possibilities to match the nodes and edges of $g_1$ to nodes and edges of $g_2$ allowing substitutions, deletions and insertions. Thus the problem of finding the minimum cost sequence of edit operations that transform $g_1$ into $g_2$ is converted into the problem of finding the minimum cost state in the search graph. Heuristics can be used to speed up the search, i.e., to avoid exploring those parts of the search graph that don't contribute to the solution [3, 4]. Regardless of any heuristics, the worst time complexity of graph edit distance computation is exponential in the size of the underlying graphs. This can be easily concluded from the fact that subgraph isomorphism detection, which is a special case of graph distance computation, is known to be NP-complete[5].

In the rest of this paper we will restrict our considerations to subgraph isomorphism detection. That is, given an actual problem $P$, represented by a graph $g$, and a number of solved problems $P_1, \ldots, P_n$, represented by graphs $g_1, \ldots, g_n$ we want to find out if any of the $g_i$ is a subgraph of $g$. Formally, $g_1$ is

[1] In the term *graph search*, *graph* refers to the representation of the underlying problem space (or state space) by means of a graph. This representation of the underlying problem space must not be confused with the graphs $g_1$ and $g_2$, the edit distance $d(g_1, g_2)$ of which is to be computed.

a subgraph of $g_2$ if all nodes and edges of $g_1$ are contained in $g_2$, and if corresponding nodes and edges have the same labels. For example, the graph shown in Fig 1c is a subgraph of the graph shown in Fig 1b. As mentioned before, determining if $g_1$ is a subgraph of $g_2$ is a special case of graph edit distance computation under particular costs of the elementary edit operations.

Subgraph isomorphism detection has a high degree of practical relevance in applications where problems are decomposable into subproblems that can be solved individually. In such an application, we would collect all previously solved subproblems in a library. Let the library be represented by graphs $g_1, \ldots, g_n$. Now given a new problem, i.e. a graph $g$, we match it to each stored case in the library. If $g$ contains one or more $g_i$'s as subgraphs then we conclude that the solutions of these subproblems can be used for the given problem.

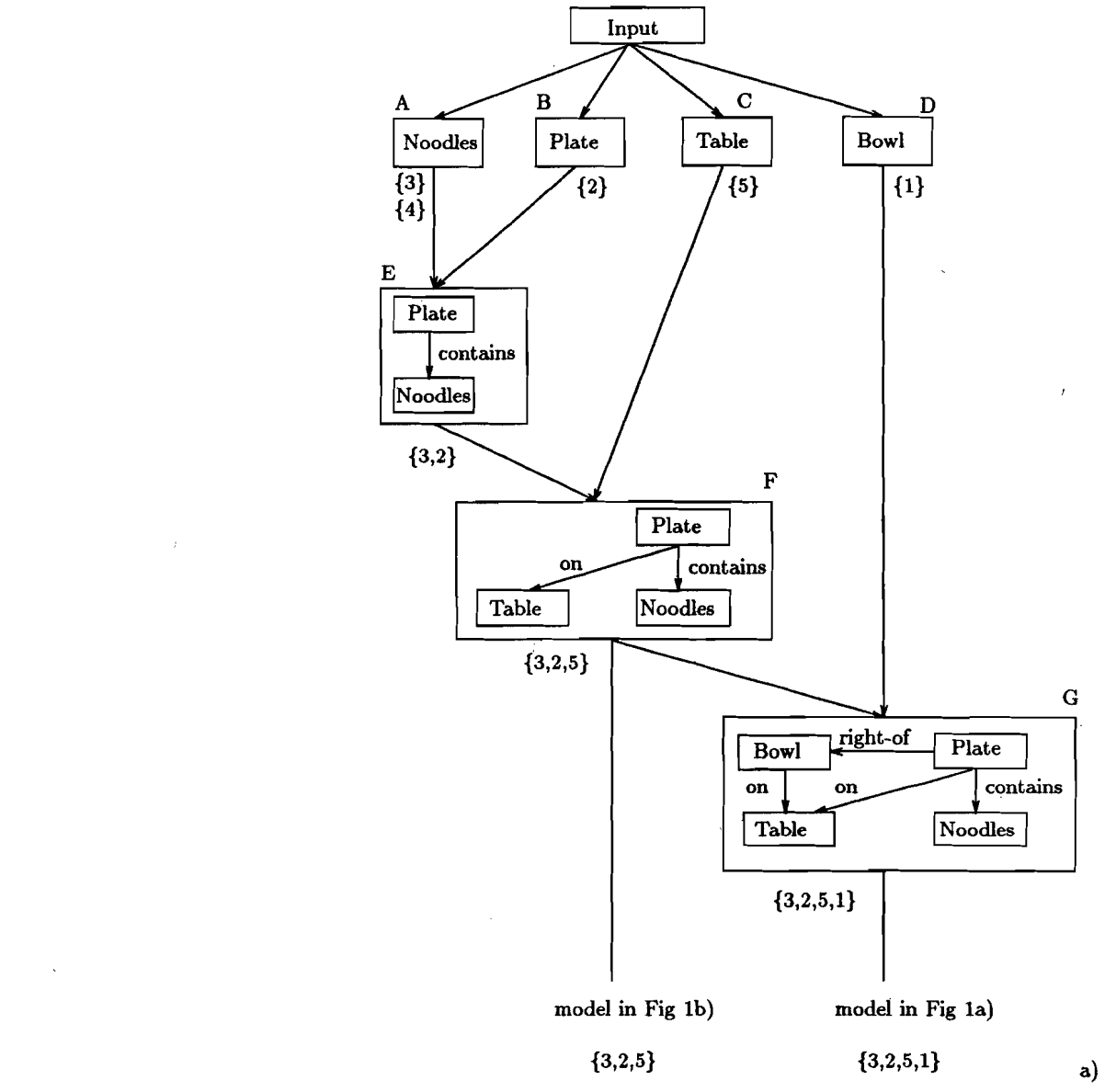
As subgraph isomorphism detection is NP-complete, we have to be concerned about computation time. The problem of computational efficiency becomes even more serious if our library of previously solved cases is large. Under a naive strategy, we would sequentially match the actual problem to each library case in order to find out if it occurs as a subproblem in the actual problem. Thus the overall computation time would increase by a factor equal to the number of cases in the library. In this paper, we propose a new method for efficient subgraph isomorphism detection. The method is particularly useful if the number of cases in the library is large because substructures that occur more than once within the same or different model graphs are considered only once by the matching procedure. Thus much computational work can be saved. It can be shown that in the limit when the model graphs become more and more similar to each other, the computational complexity of the new matching procedure becomes independent of the number of models.

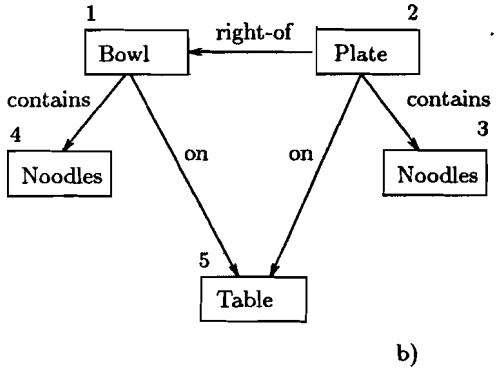
The method for subgraph isomorphism detection has some similarity with the RETE-algorithm that was introduced for efficient conflict set determination in forward-chaining rule-based systems [6, 7]. In an off-line phase, we compile the library graphs $g_1, \ldots, g_n$ into a network. This network is a compact representation of the library in the sense that nodes and edges that occur in different $g_i$'s or several times within the same $g_i$ are stored only once in the network. The network can be incrementally updated. That is, if a new solved subproblem $g_i$ is added to the library, it can be easily incorporated into the network without the need of recompiling the network from scratch, i.e., from the enlarged library.

The network for the graphs in Fig 1b and 1c is shown in Fig 2a. Due to the fact that the model in Fig 1c is a subgraph of the model in Fig 1b, any instance corresponding to the node F is considered both an instance of the model in Fig 1c, and an instance of a subgraph of the model in Fig 1b. Generally, a network like the one in Fig 2a consists of four kinds of nodes. The entrance to the network is marked by the one and only *input-node*. The input-node receives at run time the graph that is to be tested. From the input-node there are one or more outgoing n-edges[2] leading to $l$-vertex-checkers. The *$l$-vertex-checkers* are the second type of nodes in the network. The task of a $l$-vertex-checker is to test whether a vertex $v$ of the input graph has the label $l$. If the label of $v$ is $l$ then $v$ is an *instance* of any model vertex with label $l$ and is stored in a memory which is local to the $l$-vertex-checker. Each $l$-vertex-checker has one or more outgoing n-edges leading to $E$-subgraph-checkers or $g$-model-nodes (see below). An *$E$-subgraph-checker*, the third type of node, has always two parent nodes to which it is connected by one n-edge each. The parent nodes are either of type $l$-vertex-checker or $E$-subgraph-checker. An $E$-subgraph-checker represents a subgraph of one or several model graphs in the network. Its task is to find instances of this subgraph based on the instances which are found by the parent nodes. If an $E$-subgraph-checker receives a new instance from its left (right) parent then it will try to combine it with all the instances stored in the local memory of the right (left) parent. Two instances can be combined if they are disjoint, i.e., if they do not have any vertex in common, and if each edge that is specified in a list $E$ exists between the two instances. Any succesful combination results in an instance of the subgraph which is represented in this $E$-subgraph-checker. The new instance is stored in the local memory and sent to all successor nodes. There may be one or several outgoing n-edges from each $E$-subgraph-checker leading to other $E$-subgraph-checkers or to $g$-model-nodes. The *$g$-model-nodes* are the fourth kind of network nodes. For each model graph $g_i$ that was compiled into the network there is one $g_i$-model-node. Each $g$-model-node has exactly one incoming n-edge. Any instance that arrives at a $g$-model-node is an instance of the model graph $g$.

In order to explain the run time behavior of the network in detail, we consider the graph in Fig 2b. The contents of the local memories of the network nodes are displayed in brackets below the actual nodes. First, each vertex of the graph in Fig 2b is sent via the input-node to the $l$-vertex-checkers, where the vertices with the matching labels are stored in the local memory. There are two vertices, 3 and 4, in the input graph that match the label in node A. Therefore, the local memory of node A contains the instances

[2] In order to distinguish between the edges of the graphs and the edges of the network, we will refer in the following to network edges as n-edges.




b)

Figure 2: The network a) is compiled for the models in Fig 1b and 1c. The local memory contents are given after the insertion of the input graph b).

{3} and {4}. The other $l$-vertex-checkers B,C,D receive a single instance each. Next, the $E$-subgraph-checker E takes the instances found in A and B and tests whether there exists an edge labeled *contains* between them. Such an edge exists between the vertices 3 and 2, but not between 4 and 2. Thus, the instance $\{3, 2\}$ is the only one found by E. It is stored in the local memory of E. The $E$-subgraph-checker F finds $\{3, 2, 5\}$, which is an instance of the model in Fig 1c and a partial instance of the model in Fig 1b. Finally, in the node G two edge tests are performed. First, there must be an edge labeled $right - of$ from the second vertex in the left instance to the single vertex in the right instance. Secondly, there must be an edge labeled *on* from the single vertex in the right instance to the third vertex in the left instance. As both edges exist, we find the instance $\{3, 2, 5, 1\}$ in G and thus an instance for the model in Fig 1b. The algorithm terminates succesfully after both instances of the models in Fig 1b and 1c in the input graph have been found.

## 4 Computational Complexity and Experimental Results

In order to analyze the computational complexity of the proposed method for subgraph isomorphism detection, let
$N$ = number of different solved cases in the library,
$I$ = number of edges in the actual problem to be solved,
$M$ = maximum number of edges in one solved case in the library,
$M_1$ = number of edges that occur in all solved cases in the library,
$M_2$ = number of edges that are unique to each solved case, where $M_1 + M_2 = M$
The computational complexity of a naive subgraph isomorphism detection procedure that is based on graph search, treating each solved case individually , is

$$O(NM^3I^M) \text{ and } O(NM^3I) \tag{1}$$

in the worst and best case, respectively. By contrast, our proposed method has a computational time complexity of

$$O(M_1^3I^M + NM_2M^2I^M) \text{ and } O(M_1^3 + NM^2M_2) \tag{2}$$

in the worst case and best case, respectively. We notice that the two expressions in (2) become equal to $O(NM^3I^M)$ and $O(NM^3)$ for $M_1 = 0$, i.e., $M = M_2$. This corresponds to the one extreme case where there are no common parts in the solved cases in the library. Notice that in this case the worst case is equal to (1) while the best case is better than (1) by a factor of $I$. In the other extreme case, we have $M_2 = 0$ , i.e., $M = M_1$. This means that all the solved cases in the library are identical, or, in other words, the common part that is shared in the network is maximum. In this case, the two expressions in (2) become equal to $O(M^3I^M)$ and $O(M^3)$, respectively. Comparing with (1) we notice that now the time complexity is no longer dependent on the number of solved cases in the library, neither in the worst nor the best case.

The proposed method for subgraph isomorphism detection has been implemented in C++ and runs on a SUN workstation. In order to verify the results of our theoretical computational complexity analysis we run a number of experiments with randomly generated graphs. For the purpose of comparison, we also implemented a straightforward solution to subgraph isomorphism detection based on graph search and sequentially testing each of the $g_i$'s.

In the first experiment, we generated a database of 10 model graphs, each containing 50 vertices and an average of 100 edges. In order to study the influence of the size of the common subgraph on the time performance of our algorithm, we varied the size of the common subgraph of all the models between 5 and 45 vertices. For each size of the subgraph we run five test series, i.e. we generated five times a database of 10 models and measured the average time the algorithm used in order to match each model to the database. The results are shown in Fig 3. We can observe that while the tree search uses more time the larger the common subgraph becomes, our new method performs better thanks to its capability of sharing the common subgraph among the different models.

In the second experiment, we kept the size of the common subgraph constant at 20 vertices and varied the size of the database. Starting with one model we increased the number of models until 20. Each model contained a total of 30 vertices. The results of the second experiment are shown in Fig 4. The fact that for any new model added to the database a subgraph of size 20 is already represented in the network explains why the new algorithm shows only a slight increase in time for a growing database. The traditional approach, however, performs an independent matching process for each model in the database and forgets about previously found instances of the common subgraph. With 1 graph in the database, both algorithms used 0.5 seconds while in the end, with 20 graphs in the database the traditional approach took more than 3.5 seconds to terminate compared to 0.5 seconds of the new algorithm.

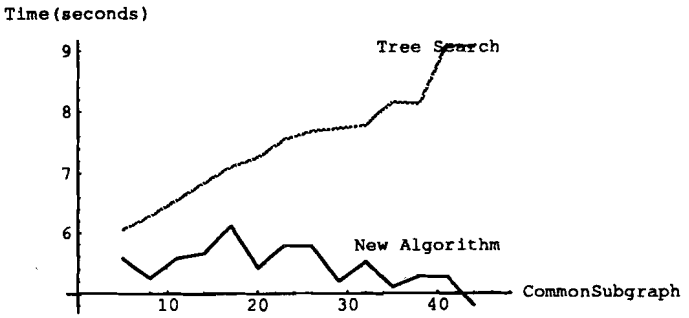



Figure 3: (first experiment) For each point of measurement we generated a database of ten models, each containing 50 vertices, including the common subgraph. The average number of edges per model was 100.

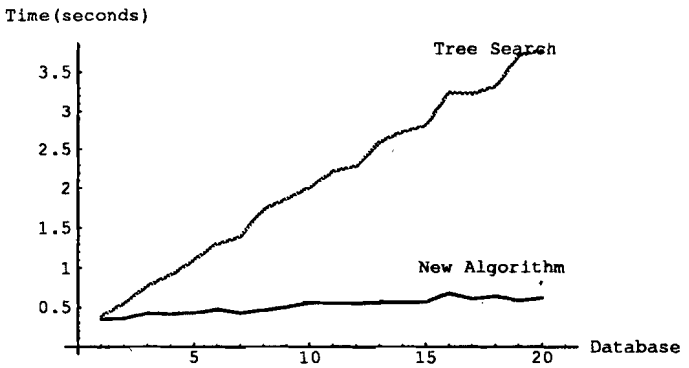



Figure 4: (second experiment) Each model in the steadily growing database contained 30 vertices including a common subgraph with 20 vertices. There was an average number of 50 edges per model.

# 5 Conclusion

The similarity of graphs is an important aspect in case based reasoning and other application areas. In this paper, we have first introduced a general framework for graph similarity based on a set of edit operations. Then, we have proposed a new computational procedure for a special case, namely subgraph isomorphism detection. Both, a theoretical complexity analysis and practical experiments have shown that the new procedure is more efficient than traditional tree search based methods for subgraph isomorphism detection. It is particularly useful if the number of cases in the library of a system is large and if the stored cases are similar to each other.

# References

[1] Richter, M.M.: *Classification and learning of similarity measures*, In Opitz, Klausen, Klar (eds.): Studies in Classification, Data Analysis and Knowledge Organisation, Springer Verlag, 1992

[2] Wagner, R.A.,Fischer, M.J.:*The string-to-string correction problem*, Journal of the ACM, Vol. 21, No.1, 168-173, 1974

[3] Ullman, J.R.:*An Algorithm for subgraph isomorphism*, Journal of the ACM, Vol. 23, No.1, 31-42, 1976

[4] Bunke, H., Allerman, G.:*Inexact graph matching for structural pattern recognition*, Pattern Recognition Letters 1, North Holland, 245-253, 1983

[5] Garey, M.R., Johnson, D.S.: *Computers and Intractability: A Guide to the Theory of NP-Completeness*, W.H. Freeman and Company, 1979

[6] Forgy, C.L.: *RETE, a fast algorithm for the many pattern / many object pattern match problem*, Artificial Intelligence Vol. 19, 17-37, 1982

[7] Lee, H.S., Schor, M.I.: *Match algorithms for generalized RETE networks*, Artificial Intelligence Vol. 54, 255-270, 1992

# System and Processing View in Similarity Assessment*

Dietmar Janetzko
University of Freiburg
D-79098 Freiburg

Erica Melis
University of Saarbrücken
D-66000 Saarbrücken

Stefan Wess
University of Kaiserslautern
D–67653 Kaiserslautern

**Abstract**

Work on similarity can be shown to follow either a system view or a processing view with the former paying more attention to architectures of similarity assessing systems and the latter concentrating on similarity metrics. As similarity depends on a number of characteristics (e.g., goals, knowledge, context, common features) both view have their own merits when assessing similarity. In this paper, we present a framework of multistage similarity assessment that provides a linkage for a modeling of similarity according to the system and processing view. In so doing, the stages of the system can be evaluated according to both the characteristics of similarity being modeled and the errors possibly made.

## 1 Introduction

During the past several years, a flurry of interest in similarity has been touched off by research done in information retrieval *(IR)*, analogical *(AR)* and case-based reasoning *(CBR)* (e.g., Vosniadou & Ortony, 1989). While puzzling out principles of similarity assessment in cognitive science and artificial intelligence two different approaches have been pursued: Investigations of similarity adopting the *processing view* strive at developing a condensed formal account of similarity intended to be used independently of the pecularities of a system's architecture. To put it another way, the core idea of the processing view has been to uncover principles of similarity as basic as possible to obtain a coverage as broad as possible. A well-known proponent of the processing view is (Tversky 1977) and his contrast model.

Conversely, research indebted to the *system view* concentrates on specifying architectural constraints on similarity assessment. That is, according to the credo of the system view characteristics of similarity may be captured by choosing an appropriate architecture of a system. Following this line, in case-based reasoning a number of models of computing similarity start with a great number of computational cheap similarity assessments. Only cases that yield a high score are taken over to the second stage to be assessed again with computationally expensive methods used to select the best scoring cases (e.g., Gentner & Forbus, 1991).

Pointing out to differences between a processing and system view is not supposed to pass unchallenged. At least when it comes down to actually building a system, so a possible caveat might go, the distinction between the two views seems to be more a difference in emphasis than in substance. Our objection to this argument is that there is quite a variety of characteristics of similarity assessment (Janetzko, Wess & Melis 1992), some of which are best modeled either according to the system view as to the processing view.

For example, the dependency of similarity assessment on the number of common and distinguishing attributes is probably best captured by the processing view. In contrast, the dependency of similarity assessment on goals, knowledge, context, or resources invested like time or memory are issues covered best by the system view. Thus, differentiating between these two views is more but a funny curiosity in the zoo of models of similarity as it can be used to guide modeling of characteristics of similarity according to the appropriate view.

The present paper is devoted to an analysis of the costs and benefits of similarity assessment according to the processing and the system view. First, the notion of process and system view is stepwisely fleshed out to gain further understanding of the possibilities given by each of both views. Second, we discuss errors that may occur within multistage similarity assessment that links the processing and the system view. Third, we introduce a three-stage model of similarity assessment and present an evaluation along with the criteria established before. Finally, we discuss relationships towards other models of similarity assessment.

*This research was supported by the "Deutsche Forschungsgemeinschaft" (DFG), "Sonderforschungsbereich" (SFB) 314: "Artificial Intelligence and Knowledge-Based Systems", projects X9 and D3.

## 2 Linking the processing and the system view on similarity

As foreshadowed by the preceeding discussion, distinguishing between the processing and the system view and differentiating among various characteristics of similarity leads to a desirable goal when building models of similarity assessment: Similarity may be best modeled in accordance to the possibilities of the two different views, i.e. on different levels. This first tenet may be called *"The principle of preferred levels of modeling"*.

Linked to that principle is another one that has to be fulfilled to make similarity assessment flexible. This second principle is referred to as *"The principle of graceful degradation"* (Norman & Bobrow, 1975). By this, we mean that similarity assessment should show a smooth decline rather than an all-or-none behavior when faced with difficulties, e.g., low-quality data or the like. This is deemed important if resources (e.g., time, memory) are limited or if the system itself does intend to limit resources (e.g., to perform a preselection) in order to invest resources in an economical fashion.

Finally, the principle just mentioned implies a third one. This is called *"The principle of continually available output"* (Norman & Bobrow, 1975; Russell & Zilberstein, 1991). To spell out this principle is to specify the principle of graceful degradation. As a consequence, it should be possible to stop processes of similarity assessment, e.g., by retracting resources needed, and obtain results that are usable by the system although suboptimal when compared to the results acquired without stopping similarity assessment.

The ideas in this paper rely on the conjecture that modeling of similarity assessments according to the three principles mentioned above is only possible by linking the processing and the system view. When put into practice, the principle-guided linkage of the two views amounts to a multistage similarity assessment with characteristics of similarity brought into focus by each view distributed on different stages. The framework of such an architecture provides a number of advantages: Modeling of characteristics of similarity can and should be done on different stages according to the principle of preferred levels of modeling. Depending on the stage of processing reached there is a smooth decline in the quality of the system's output, which obeys to the principle of graceful degradation. Finally, an architecture of multistage similarity assessment allows for a good approximation to the principle of continually available output as each stage may serve an exit-point for similarity assessment. The quality of the similarity assessment reached at each exit-point is a function of the resources invested.

## 3 Demands on the assessment of similarity

In what follows, we characterize two basic requirements to be fulfilled when assessing similarity. This is done along with a discussion of how to put the ideas of this paper into practice when building a system and an eye towards related work in information retrieval, analogical, and case-based reasoning. In so doing, we will find further evidence for a multistage similarity assessment, which is spelled out in subsequent sections.

### 3.1 Efficiency

Analyzing the process of similarity assessment from a efficiency point of view results in the demand of low computational costs of the retrieval. Since all items of the knowledge base are involved in the first step of the process, it is reasonable to require the first step to work very quickly on each item. The next step which works already on a set of preselected cases may have higher relative costs.

A similar goal is aimed at by open hashing in databases: The hash function makes it possible to access - a list of items very fast; the search within this list, being as short as possible, has higher relative costs.

In database research a lot of other retrieval approaches has been developed that are computationally cheap e.g., multidimensional associative binary trees, called *k-d Trees* (Bently, 1978), close match retrieval (Friedman, Bently & Finkel, 1977), incremental nearest–neighbor search (Broder, 1990), best–match retrieval based on Voronoi-Diagrams (*c.f.* Mehlhorn, 1984) or hypercubes (Stolter, Henke & King, 1989). These techniques are able to retrieve a best–match based on a set of surface features in logarithmic expected time $O(log(n))$ where $n$ is the number of stored items in the database.

The now commercial available case-based reasoning shell REMIND (Cognitive Systems, 1991) developed by Cognitive Systems an enterprise founded by R.C. Schank uses this kind of rapid retrieval algorithms for case-based reasoning.

Other approaches to a computationally cheap search of similar cases use the assessment of similarity on the basis of the dot product over feature vectors (Medin & Schaffer, 1978), connectionist models of learning (Rummelhart & McClelland, 1986), the PATDEX–approach (Wess, 1991; Richter & Wess, 1991),

or the memory-based reasoning approach (Stanfil & Waltz, 1986), which relies on a massive parallel search on a connection machine.

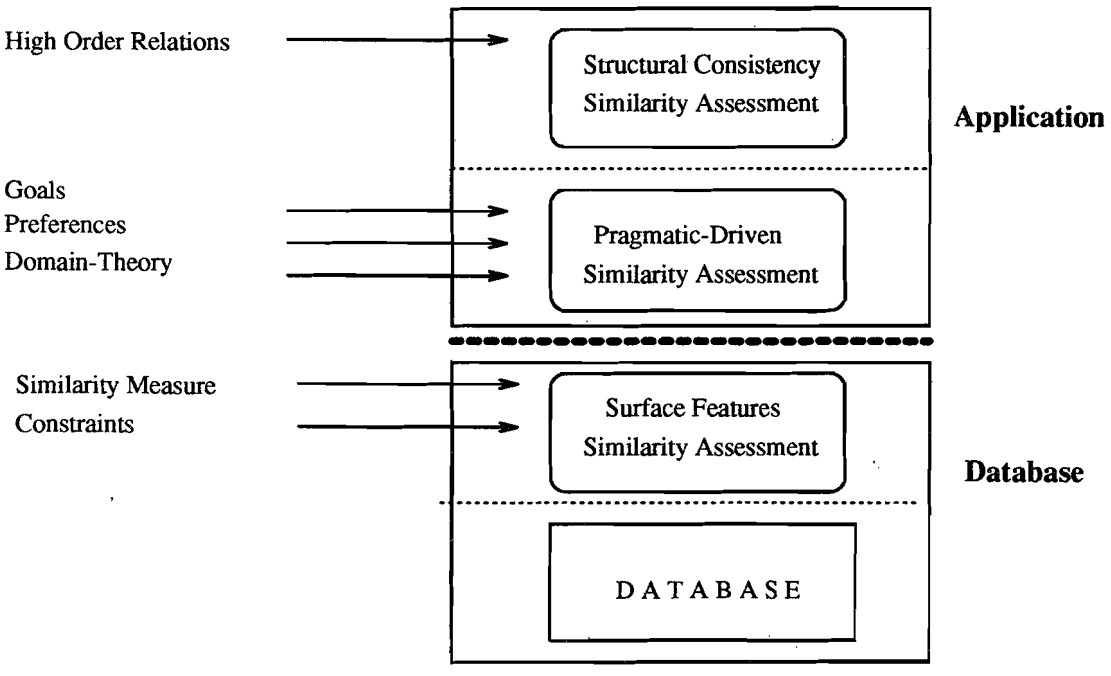



Figure 1: The system view

## 3.2 Reliability

The analysis of the process from a reliability point of view yields characteristics of the kind of errors occurring at the consecutive steps of the process. These types of errors are well known in statistics as they give an account of the errors that can be made whenever a hypothesis is accepted or rejected (e.g. Bock, 1975). We use the notion of $\alpha$-error and $\beta$-error to classify possible errors to be made when assessing the similarity between two cases.

**Definition 1 ($\alpha$-Error)** *If a previous case being useful to solve a problem at hand is part of the case base but not selected, the error is called $\alpha$-error.*

**Definition 2 ($\beta$-Error)** *If a previous case not being useful to solve the current problem is part of the case base but selected, the error is called beta-error.*

Each model of selecting cases has to account for both kinds of errors. The selection of similar items (e.g., cases, concepts, entries in a database) is guided by selection criteria. $\alpha$- and $\beta$-errors depend on the selection criteria applied to find similar items. Selection criteria causing no $\alpha$-error are necessary criteria, and selection criteria causing no $\beta$-error are sufficient criteria.

As well known (Mitchell, Keller & Kedar-Cabelli, 1986), explanation-based generalization *(EBG)* provides sufficient descriptions. The goal-driven similarity assessment in (Janetzko, Wess & Melis, 1992) using the EBG-method provides sufficient criteria and tends to keep the $\beta$-error low.

The ideas that form the basis of $\alpha$– and $\beta$-errors are closely related to the version space method introduced by (Mitchell, 1982). The description to follow shows how the version space technique can be applied to find a selection criteria that keep $\alpha$– and $\beta$-error at the lowest level possible.

Let the example space be a set of pairs of items. The criteria space $CRIT$ is taken to mean a space of formulae representing selection criteria, i.e., the analogue to Mitchell's concept space. The partial order on $CRIT$ (*more specific, more general*) can be defined analogously to the hierarchy of generalizations in the version space. In agreement with the version space method the search space $CRIT$ is reduced from top and from bottom introducing $G$ (as the set of *most general criteria* selecting all known positive examples and rejecting all known negative examples) and $S$ (as the set of *most specific criteria* selecting all known positive examples and rejecting all known negative examples).

The criteria from $G$ keep the $\alpha$– and the criteria from $S$ keep the $\beta$-errors at the lowest level possible. Following Mitchell's model, if $G = S$ the concept is learned and no $\alpha$– or $\beta$–errors occurs.

It is desirable that in the *first step* of case selection no (or almost no) $\alpha$-error occurs, that is, no useful previous case is excluded from further processing. It is also desirable that in the *last step* cases being not useful are excluded.

There are several possibilities to meet this demand: If during the first step of the process no $\alpha$-error occurs, and there are useful items in the database then there remains a nonempty set $\mathcal{C}$ of items. During the next steps from $\mathcal{C}$ items may be selected the computational costs of adaptation are lowest for. Items with computational costs of adaptation that exceed those to be expected can be eliminated.

There are several approaches including such a usability assessment. For example, the goal-driven similarity assessment (Janetzko, Wess & Melis, 1992), similarity conserving transformations (SCT's Koton, 1988) evaluate certain similarities and dissimilarities as relevant or irrelevant.

# 4 Stages of similarity assessment

As noted earlier, a multistage similarity model has been deemed necessary to cover a number of issues involved in similarity assessment. Among the most important of those issues are the possibility to combine various models of similarity assessments according to different characteristics of similarity. In this way, it is feasible to control the impact of each of those characteristics. Additionally, multistage similarity assessment allows for specifying constraints on errors such that the $\alpha$–error should be low in the *first* and the $\beta$–error should be low in the *last* stage. During the stages the number of items considered decreases and the computational costs per item increase.

**Stage I - Syntactic features:** Multistage similarity assessment begins by using a syntactic measure of similarity which is based on features that form an explicit part of the representation of the items being compared. Measures deriving similarity from the number of common and different features that may or may not be combined with weigths can be used at this stage (Tversky, 1977). Alternatively, models of similarity assessment mentioned in 3.1 like k-d trees may be employed for that purpose. At this stage, similarity assessment does not depend on the representation of the domain theory. No knowledge but that encoded in the items (cases, entries of database) is used explicitly. Since this stage is usually computationally cheap it is well suited to be used for preselecting items.

**Stage II - Pragmatic relevance:** A pure syntactic approach is not sufficient for similarity assessment. First, a difference with regard to only one feature results in a high statistical similarity score but may be based only on a high agreement with regard to unimportant features. Vice versa, a great number of differences between two cases leads to a poor statistical similarity score but may camouflage an agreement with regard to important features. For that matter, the next stage proceeds by allowing for the influence of pragmatic determinants (e.g., goals and knowledge) on similarity assessment. In goal-driven similarity assessment (Janetzko, Wess, & Melis, 1992), for example, a set of features is computed by using EBG to single out those features that are of pragmatic relevance according to a goal and a domain theory. At this stage, similarity assessment makes use of the representation of the domain theory and pragmatic determinants like goals or purposes. This stage is computationally more expensive than the first one.

**Stage III - Consistency:** For economical reasons, the kind of knowledge used in multistage similarity assessment is distributed on three stages. Knowledge that can be used as a test to rule out similarity of items has not been employed in the preceeding stages. This kind of knowledge is taken to reject items that are definitely dissimilar when compared to the input item. This stage is extremely dependent on the domain theory and on the application. As a result, there are various possibilities to perform consistency tests. For example a diagnostic application consistency may be defined by a model–based diagnosis approach *c.f.* (Koton, 1988). Depending on the respective application this consistency check may be very expensive. Hence, this procedure is left for the last stage of similarity assessment.

# 5 Conclusions

Although up to now there is not a clear division into demands for knowledge-based steps of retrieval of cases and others, empirical results show a correspondence of knowledge-based and not-knowledge-based preselections with the selection of cases by experts and novices respectively.

Novick (1988) has found differences between the retrieval cues available for the retrieval process by experts and novices: Novices almost exclusive rely on salient surface features of the target. Experts, however, will be able to use both surface and structural features. For common domains Holyoak and Koh (1987) established that retrieval of analogues relies more on surface similarity and less on structural similarity (than mapping). This might be simulated in the retrieval included in CBR by a pure statistical

preselection followed up by a more knowledge-based final selection step. An attempt to capture the novice pheneomenon is done by Gentner and Forbus (1991). They use as a first stage a matcher that works as follows: Each case is stored with a content vector (vector of number of occurrences of predicates, functions, and connectives) The content vector of each case is compared with the computed content vector of an entered probe. Hence, this stage consists of a purely statistical syntactic comparison. Afterwards a matcher calculating literal similarity is applied to the output of the first stage.

This does not mean that knowledge-based similarity assessment in general provides only sufficient selection criteria. On the contrary, the domain theory can provide necessary criteria, too.

Depending on the pecularities of the domain there is the possibility to introduce knowledge-based modifications, e.g., of a pure statistic preselection by the contrast rule (Tversky & Gati 1982). This may be reasonable if the domain under study provides features or combinations of features which make usability probable or which rule out usability.

# 6 References


**Bently, J.L. (1975).** Multidimensional Binary Search Trees used for Associative Searching. Communications of the ACM, 18(9), 509- 517.

**Bock, R. D. (1975).** Multivariate statistical methods in behavioral research. New York.

**Broder, A.J. (1990).** Strategies for efficient incremental nearest neighbor Search. Pattern Recognition, 23(1), 171-178.

**Cognitive Systems (1991).** REMIND Solutions from prior experience. Technical Report, Cognitive Systems, Inc., MA, USA.

**Friedman, J.H. Bently, J.L. & Finkel, A.F. (1977).** An Algorithm for Finding Best Matches in Logarithmic Expected Time, ACM Transactions on Mathematical Software, 3(3), 209-226.

**Gentner, D. & Forbus, K.D. (1991).** MAC/FAC: A model of similarity-based retrieval. Proceedings of the Thirteenth Annual Conference of the Cognitive Science Society, Chicago, Ill, 504-509.

**Holyoak, K.J. & Koh, K. (1987).** Surface and structural similarity in analogical transfer. Memory & Cognition 115, 332-340.

**Janetzko, D. Wess, S. & Melis, E. (1992).** Goal-Driven Similarity Assessment, In: Ohlbach (Ed.), German Workshop of Artificial Intelligence 92, Springer Verlag.

**Koton, P. (1988).** Reasoning about evidence in causal explanations. In: J. Kolodner (Ed) Proc. of a Workshop on Case-Based Reasoning, Florida. Morgan Kaufmann, 260-270.

**Medin, D.L. & Schaffer, M.M. (1978).** Context theory of classification learning. Psychological Review 85 (1978), 207-238.

**Mehlhorn, K. (1984).** Multi-dimensional Searching and Computational Geometry, Springer.

**Mitchell, T.M. (1982).** Generalization as search. Artificial Intelligence 18, 203-226.

**Mitchell, T.M. Keller, R.M. & Kedar-Cabelli, S.T. (1986).** Explanation-based generalization: A unifying view. Machine Learning 1,47-80.

**Norman, D.A., & Bobrow, D.G. (1975.** On data-limited and resource-limited processes. Cognitive Psychology,7, 44-64.

**Novick, L.R. (1988).** Analogical transfer, problem similarity and expertise. Journal of Experimental Psychology, Learning, Memory , and Cognition 14, 510-520.

**Puppe, F. & Goos, K. (1991).** Improving case based classification with expert knowledge. In: Th. Christaller (Ed.), German Workshop of Artificial Intelligence 91, 196-205, Springer Verlag.

**Richter, M.M. & Wess, S. (1991).** Similarity, Uncertainty and Case-Based Reasoning in PATDEX. Kluwer Academics.

**Rummelhart, D. E., & McClelland, J. L. (1986).** Parallel distributed processing. Cambridge, MA: MIT Press.

**Russel, S.J. & Zilberstein, S. (1991).** Composing Real-Time Systems. Proc. IJCAI-91, 212-217.

**Stanfil, C. & Waltz, D. (1986).** Towards memory-based reasoning. Communications of the ACM 29, 1213-1229.

**Stolter R.H. Henke A.L. & King J.A. (1989).** Rapid Retrieval Algorithms for Case-Based Reasoning, Proc. IJCAI-89.

**Tversky, A. (1977).** Features of similarity. Psychological Review, 84, 327-352.

**Tversky, A. & Gati, I. (1982).** Similarity, separability, and the triangle equality. Psychological Review 89, 123-154.

**Vosniadou, S. & Ortony, A. (1989).** Similarity and analogical reasoning. Cambridge University Press.

# SIMILARITY ASSESSMENT AND CASE REPRESENTATION IN CASE-BASED DESIGN

Markus Knauff & Christoph Schlieder
University of Freiburg
Center for Cognitive Science
D-7800 Freiburg, FRG
[knauff][cs]@cognition.iig.uni-freiburg.de

**Abstract**

In this paper we are engaged in the interpretation and representation of spatial arrangements and relationships that hold between different kinds of objects, because this is one of the most crucial issues in case-based building design. The core idea of this paper is, that what we represent in a case is not only an abstraction of the spatial arrangements, but also an abstraction of how a case was built. Every case represents two different kinds of information: The first abstraction concerns the shape and location of object and their spatial relations. We present a *relational symbolic case representation* that takes primarily this information into account. The second abstraction concerns the *constructive-technical information* of a case, therefore information about the applied sequence of spatial operators. This information is represented in an *operational symbolic case representation*. Thus, we discuss how useful are information about the sequence of applied operators to solve *ambiguity problems* that are connected to a symbolic reinterpreation of a spatial arrangement and model a *change of perspective* taken this operator sequences into account.

## 1 Background and motivation

The interpretation and representation of spatial arrangement and relationships between different kinds of objects belongs to the most crucial issues in case-based building design. Our work in this context is influenced by two approaches that were presented by [Gentner1983] on the one hand and [Indurkhya1992] and [O'Hara1992] on the other.

In her *structure-mapping theory* Gentner [Gentner1983] [Gentner and Forbus1991] argues - as opposed to other approaches - that the meaning of a given case has to be derived from the relations of its parts, rather than from attributes or properties. A case is described by higher order relations between its parts, and similarity assessment is based on these relations. A lot of psychological experiments indicate how powerful this approach is and the main idea is very important from a practical point of view as well as from a cognitive science perspective.

However, in recent years in particular [Indurkhya1992] and [O'Hara1992] emphasize that Gentner's approach only deals with *fixed-descriptions* of a case. As according to [Indurkhya1992] in the context of analogy it is very important to take into account that humans often take different points of view when assessing similarity. The key idea is a process by which new points of view can be created and these redescriptions can be useful in the matching process. This criticism is meaningful from a cognitive science perspective and is just as important as Gentner's *structure-mapping approach*. Thus, it is appropriate to combine these two approaches.

What [Indurkhya1992] and [O'Hara1992] also contribute to this issue is to emphasize the role of operator knowledge in analogical reasoning. In his system PAN, [O'Hara1992] proposes besides a set of objects, a set of one-argument operators like `TRANSLATION`, `ROTATE`, `REFLECT`, `SCALE` and a two-argument operator, `GLUE`. The key idea is to represent the shape of figures taking these operators into account and similarity assessment is modeled as a match of operators. Indurkhya's approach was supported from a cognitive psychology perspective by [Knauff and Schlieder1993]

and [Boerner1993] proposes an approach in the domain of architecture design, that is strongly influenced by such approaches.

Strictly speaking, the main concern of Indurkhya's and O'Hara's approaches is shape representation and they thereby deal with spatial relations among objects. On the other hand, the main advantage of Gentner's approach lies in the stressing of the importance of the relations between the parts of a case for similarity assessment. In the light of this, we concentrate in this chapter on the combination of operator representation and spatial relations.

It should be noted here that a symbolic description of a spatial arrangement can be either based on *relations* or on *operators*. The first type of description specifies spatial relations that hold between the objects. A typical example would be to represent the distance between two objects by `distance(A,B,50)`. The other type of description specifies how the arrangement is constructed out of a set of primitive objects by applying spatial operators. As part of such a description the expression `B = translate(A,50)` could appear which says that the object `B` is obtained by translating the object `A` a certain distance.

We will distinguish these two approaches to spatial representation and speak about *relational* and *operational symbolic descriptions*. By making appeal to different mathematical concepts, relational and operational description express a difference in emphasis too, i.e. emphasis on the static (perceptive) or the dynamic (constructive) aspect of spatial arrangements respectively. The core idea of our work is to integrate these two aspects in the process of spatial similarity assessment.

We present an approach that represents cases symbolic as sequence of *primitive and complex spatial operators*. A *primitive spatial operator* in our meaning combines or moves primitive parts such as COPY, MOVE. A complex spatial operator is a combination of a set of primitive operators, which application leads to a specific spatial arrangement. The *complex spatial operator* ARRANGE-IN-A-LINE for example, builds a line of a variable number of objects (may be with different shapes). Possible parameters are `x-orientation, y-orientation, number-of-objects, distance-of-objects`. The following example gives you a first idea of what we are planning to do. It illustrates the function of a very important operator, which can be called ARRANGE-PARALLEL-TO-LINE.

## 2 A short glimpse at a typical planning problem in FABEL

In the University of Karlsruhe, the computer-driven construction-design system DANCER has been developed. The system helps architects deal with complex design-and planning- processes and enables them to present designs in a more transparent and comprehensive manner. The origins of this system are the architectual works of Fritz Haller, in particular the construction set for multistorey office or school buildings, called MIDI. It intergrates the complete technical equipment with cabeling, piping warm and cold water, used and fresh air, electric supplies and so on. [Hovestadt1992]. In DANCER all subsystems are represented as circles and ellipses, which denote places in a multidimensional design space. To keep it simple we outline one typical example of a planning situation.

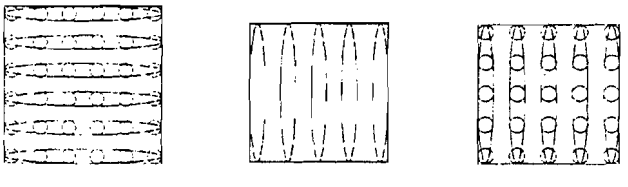


Figure 1: (a) Given Solution; (b) New Problem; (c) Solution of the new planning problem

Figure 1. shows a part of the ellipse and circle representation of an air network of an office building. Figure 1(a). describes a known arrangement of supply air outlets for a given building, which had been planned earlier. Figure 1(b) is a new problem description: The architect has to plan the arrangement of the outlets of a fresh air network in another building and the ellipses represent only their rough location. As we immediately see, 1(a) is similar to 1(b), because there are only three differences: The orientation of the network, the difference between fresh and supply air and the size of the building. Figure 1(c) shows the solution of this problem, by adopting the given example 1(a).

Formulated in generally applicable terms, a case, in our approach, is given as a sequence of diagrams - the first represents the starting point or problem description, the last, the goal or solution description. Figure 2. is absolutely simplified, but it shows the double application of the operator ARRANGE-PARALLEL-TO-LINE and the result.

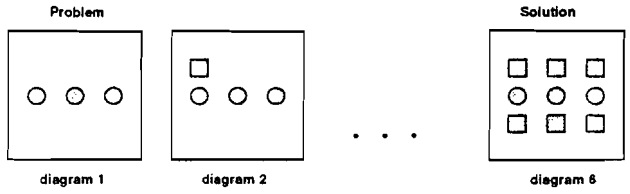



Figure 2: Case in case library

We are going on the assumption now that this case is represented in the case library and another problem description as in figure 3. is given.

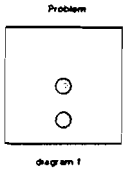



Figure 3: New problem

The new problem is *similar* to the represented case, because it can be solved by the application of the same spatial operator ARRANGE-PARALLEL-TO-LINE. The only difference to the represented case is the x- or y-orientation and the number of objects. We now just have to adopt this solution taking these two differences into account. Two parallel new lines will be built (figure 4).

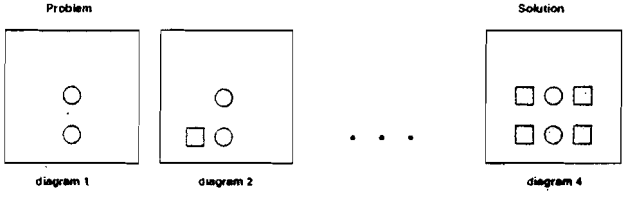



Figure 4: Solution of the new problem

## 3 Using information about the perception of a case and how it was built: Relational and operational case representation

It is important to recognize the core idea of this paper: What we represent in a case is not only an abstraction of the spatial arrangements, but also an abstraction of how a case was built. Every case represents two different kinds of information:

- The first abstraction concerns the shape and location of object and their spatial relations and the *relational symbolic case representation* takes primarily this information into account.
- The second abstraction concerns the *constructive-technical information* of a case, therefore information about the applied sequence of spatial operators and the *operational symbolic case representation* takes primarily this information into account.

We will call the first (perceptive) abstraction *grouping*, the second (constructive) *sequencing*. The following figure gives a flavor of this distinction.

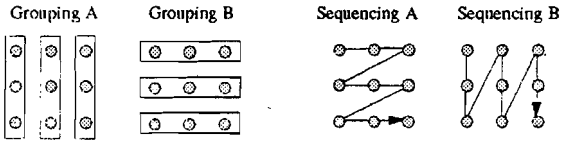



Figure 5: Grouping and sequencing of the same spatial arrangement of objects

Formulated in generally applicable terms the *relational object representation* (REL) represents the groupings of the objects of a case as a *partition* of the set of objects, whereas the *operational object representation* represents the sequencings as a permutation of a set of objects and we know that there are much more permutations than partitions [1] We have exactly

$n!$ sequences, in our case 362880, if n=9

$2^n$ groupings, in our case 512, if n=9

As we immediately see, we get more information from sequencing, because it chooses from much more alternatives. Our proposal is to represent these two different aspects of a spatial arrangement in the concept-network. Thus, every concept represents firstly information about the arrangement of objects, and secondly one or just a few default construction sequences. The concept `arranged-in-a-line`, for example, can be captured in the following figure.

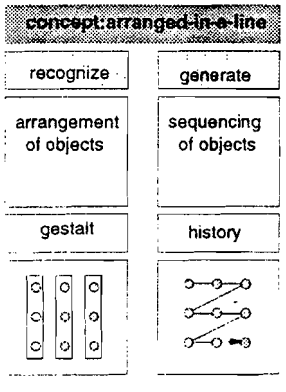



Figure 6: Components of the concept `arranged-in-a-line`

We call that part of the concept representation that represents the arrangement of objects the *recognize-component*, while the representation of defaults sequences is called *generate-component*. The *recognize-component* must match a parametrized geometrical case representation with a symbolic representation of a concept. The *generate-component* has to transform a symbolic representation into a geometric representation. The following figure gives you a brief impression of *relational object representation* and the *operational case representation* of the same case.

[1] A *partition* of a set M is a disjoint union of subsets. A *permutation* of n distinct objects of length k is an ordered arrangement of any k of the objects [Graham *et al.*1989].

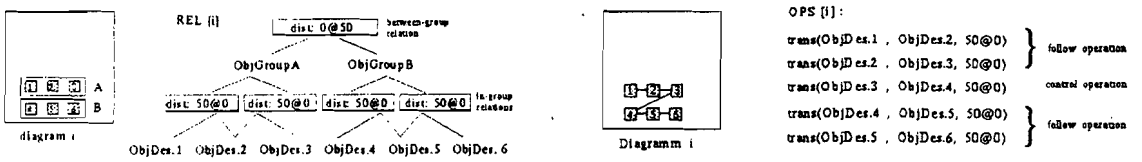



Figure 7: The relational and operational case representation format

# 4 Ambiguity and change of perspective

A key issue of a symbolic reinterpretation of a geometric representation of a spatial arrangement is its *ambiguity*. That means that there are a lot of different ways of interpreting a given spatial arrangement and we have to find the right - or at least the best- one. Gentners approach has to deal with such problems and Indurkhya's and O'Hara's works on *representational change* are concerned with this issue too. We now want to use the above ideas to deal with such problems and propose a *change of perspective*, taking different operator sequences into account.

In principle, there are as many perspectives on a case as permutations of the objects are possible. In order to better understand the process by which people construct a spatial arrangement of objects we are planning to conduct a series of psychological experiments. Special attention will be paid to the sequencing of spatial operators such as moving, copying and combining object parts. Our working hypothesis predicts that the number of ways in which humans actually sequentialize the construction process is far inferior to the number of all combinatorially possible operator sequences. Analysis of the performance data should further show whether a set of preferred operator sequences can be identified, that is, sequences which people use as standard, default solutions when they are confronted with a construction task.

If evidence supports the assumption of default sequences this would provide a natural solution to an issue that [Indurkhya1992] addresses in connection with his computational approach to analogy. Central to any analogy problem of the type A is to B as C is to D (where D has to be computed given A, B and C), is the representation of the objects involved. In the spatial domain the object representation specifies how an arrangement can be constructed out of primitive parts by means of primitive operators. Generally, the representation is not uniquely determined because there are different ways to decompose an arrangement into parts. Since the computational approach to analogy is based on structure-preserving mappings between object representations this kind of indeterminacy becomes a serious obstacle. There is no escape from the problem by simply restricting the number of primitive parts and operators. As [Indurkhya1992] pointed out indeterminacy can arise with just a single primitive part and two unary operators.

[O'Hara1992] proposed to resolve the problem of representational indeterminacy by introducing a normal form for operator-based descriptions of spatially arranged objects. His PAN algorithm uses normal form input descriptions of geometrical analogy problems. This dramatically simplifies the task of finding an analogy at least from the point of view of computational complexity. The essential idea behind this definition of the normal form is to distinguish between an operator GLUE that combines primitive parts and the operators that move primitive parts, such as ROTATE. A normal form representation then consists of an operator tree whose top node is a GLUE operation possibly followed by other GLUE operations in right-associative form. The movement operators appear as inner nodes without their position obeying any further constraints. However, as [O'Hara1992] observed, this normal form does not eliminate all representational indeterminacy. Some spatial arrangements can be described by different normal form representations. It is questionable whether the choice of the label "normal form" is appropriate under these circumstances.

Default sequencings of operators that people use when constructing spatial arrangements could provide a better solution to the problem of representational indeterminacy. Consider a spatial arrangement consisting of an alignment of objects, say, five squares in a horizontal row. We conjecture that there is only a very restricted number of default ways to construct such an arrangement (1-2-3-4-5, 5-4-3-2-1, 1-5-2-3-4, ...?). It would make sense to take only these standard decompositions into account when building a description based on a single primitive part, the square, and a unary operator, translation. One would not end up with a single normal form, but instead with a small set of standard representations. We do not expect the number of standard representations to be large since cognitive processes typically only consider a constant number of cases out of an exponential number of possible ones. A further reduction of complexity could be achieved by ranking the standard representation according to its frequency of use as can be revealed by psychological experiments.

## 5 Acknowledgements

This research was supported by the German Ministry for Research and Technology (BMFT) within the joint project FABEL under contract no. 413-4001-01IW104. Project partners in FABEL are German National Research Center of Computer Science (GMD), Sankt Augustin, BSR Consulting GmbH, München, Technical University of Dresden, HTWK Leipzig, University of Freiburg, and University of Karlsruhe.

This paper is inspired by this project but it presents our private opinion and it is clear that it does not necessarily reflect the mainstream opinion in our group. We would like to thank all collegues in particular Dietmar Janetzko and Gerhard Strube for helpful discussions and support.

## References

[Boerner1993] K. Boerner. Structural similarity as guidiance in case-based building design. *submitted to 1st EWCBR*, 1993.

[Gentner and Forbus1991] D. Gentner and D. Forbus. MAC/FAC: A model of similarity based retrieval. In *Proceedings of The 13th Cognitve Science Conference*, pages 504–509, 1991.

[Gentner1983] D. Gentner. Structure-mapping: A theoretical framework for analogy. *Cognitive Science*, 7, 1983.

[Graham *et al.*1989] R. L. Graham, Donald E. Knuth, and O. Patashnik. *Concrete Mathematics*. Addison Wesley, Reading MA, 1989.

[Hovestadt1992] L. Hovestadt. A4-Digitales Bauen. Technical report, Institut für Baugestaltung Baukonstruktion und Entwerfen der Universität Karlsruhe, Germany, 1992.

[Indurkhya1992] B. Indurkhya. *Metaphor and Cognition*. Kluwer Academic Publishers, 1992.

[Knauff and Schlieder1993] Markus Knauff and Christoph Schlieder. How to combine bottom-up and top-down processes in spatial similarity assessment? In *Approaches to similarity assessment in FABEL: FABEL-REPORT No. 13*, chapter 10. GMD, Sankt Augustin (in press), 1993.

[O'Hara1992] S. O'Hara. A model of the redescription process in the context of geometric proportional analogy problems. In K.P. Jantke, editor, *Analogical and Inductive Inference - International Workshop AII92*, pages 268–293. Springer-Verlag, Berlin, 1992.

# Applications of Case Based Reasoning to the Law the Problems of Multiple Case Reasoning and Indexing.

Mohammadali Montazeri
Alison E. Adam

Department of Computation, UMIST
P.O. Box 88 Manchester M60 1QD, UK
email: mm@sna.co.umist.ac.uk
email: a_adam@mac.co.umist.ac.uk

## 1 Introduction

The application and use of case law provides a fertile area for testing concepts and ideas from the domain of case-based reasoning. In UK law, the legal decision maker is as much bound by past case law as by statutes. It is the job of the legal decision maker to decide whether a case in case law matches the present case in order that the present case should be decided in same way as a relevant past case. Acts which have been on the statute book for some time may have generated significant bodies of case law, all of which may be potentially relevant. Complex domains such as legal reasoning require the ability to choose between and combine exemplar-based reasoning and generalization so that the techniques can be used in support of one another. Index transformation can also provide a different view of the case base by leading the problem solver to previously inaccessible cases. The present research looks at strategies where past cases may be indexed and matched to the present case and ultimately where solutions may be adapted to suit the present case. Multiple case, multiple features retrieval is proposed an important strategy for the retrieval of past cases. Generalized features, exemplar-based reasoning and indexing are incorporated into the design of the prototype system. UK employment law has been chosen as suitable application area.

## 2 Multiple Case, Multiple Features Retrieval

There are a number of specific areas of interest when looking at the use of CBR in the law. A legal decision maker may use part of a solution from a past case or a number of partial solutions from past cases to reason about the present case. The problem here involves both the representation of previous past cases, where important features must be represented in hierarchical order and also the adaptation of several partial solutions to form an overall solution or suggested decision. The retrieval of multiple cases based on multiple features is an important strategy. Most case-based reasoning systems use a single 'best' or 'most similar' case as the basis for solution [2,5,10], but clearly systems most suited to legal reasoning are those which combine pieces of several old cases to solve a new problem [1,4,12,13,14].

Our approach to multiple features retrieval strategy involves using a two step algorithm. First by using generalized features, the system retrieves all cases which match or partly match the target case. Then specific features are compared between the retrieved cases in order to choose the most similar case. In the second step, index transformation can be used to define specific features. Suppose the case base has cases relating to employment law, with important features described as below.

- *Employment, Sex discrimination, Indirect discrimination.*
- *Employment, Sex discrimination, Direct discrimination.*
- *Employment, Sex discrimination, Indirect discrimination, Part-time workers.*

- *Employment, Sex discrimination, Occupational pensions.*
- *Employment, Sex discrimination, Indirect discrimination, Equal pay.*
- *Employment, Sex discrimination, Indirect discrimination, Equal pay, Special Circumstances.*

If the case base is very large and has covered several different areas of law, then the feature *Employment* can be used otherwise it can be eliminated. To retrieve cases relating to Sex Discrimination Law we can use *Sex discrimination* as a generalized feature and retrieve all similar cases. If we add *Indirect discrimination* to the generalized previous feature to make it more specific, we will have fewer cases but they are more similar. By using this algorithm considering more specific features like *special circumstances*(e.g. *Material difference*) we can retrieve the most similar case. If no exact match occurred when using specific features, we can always backtrack and pick up cases from the previous search. In the above format the first feature of the list is the most general feature and the last one is the most specific one. For example, the case of FLETCHER V.CLAY CROSS (QUARRY SERVICES) LIMITED can be indexed by using *Sex discrimination, Equal pay, Material difference.* The first two features are very common, so to find more similar cases we use a *Special circumstances* feature like *Material difference.* By doing this the following cases will be retrieved.

- E. COOMES (HOLDINGS) LTD. V. SHIELDS [1978] I.R.L.R. 263 (C.A.). FEATURES(*Sex discrimination. Equal pay. Material difference. Extrinsic forces*).
- NATIONAL COAL BOARD V. SHERWIN AND SPRUCE [1978] I.R.L.R. 122 (E.A.T.). FEATURES(*Sex discrimination. Equal pay. Material difference. Extrinsic forces*).
- HODGSON V. J.M. FIELDS INC.[1971] 335 F. SUPP.731. FEATURES(*Sex discrimination. Equal pay. Material difference. Market forces*).
- BERNNAN V. CITY STORE [1973] 479 F.2D 235. FEATURES(*Sex discrimination. Equal pay. Material difference. Market forces*).

All four cases above are similar to the Fletcher case and can be used as an exemplar to reason by. Those cases also can be retrieved by using the *like work equalpay* feature.
Generalized features, such as we describe above, are conceptually simple but this must not disguise the necessity of including such a mechanism in a system of this type and must also not obscure the need to describe generalized features in such away that the system can use them meaningfully.

## 3 The Use of Exemplars in Legal Reasoning

The approach taken in our research acknowledges and is designed to work around some important inherent limitations of the law. Legal rules are of necessity underdetermined in that they can never be written in such a way as to categorize all areas where they should apply; the law is made by the process of interpretation and decision making; what Hart, the philosopher of jurisprudence, [6], described as the "open textured" nature of the law. A domain of open-textured rules can be partially defined by examples. The advantage of legal reasoning is that the court cases are recorded and published, and can be used to provide a set of facts. Because of their open textured nature, legal rules can cover a wide range of possibilities without any specification. The same example can be used differently in two different cases. For example the words *equal pay* as an open textured concept can cover many different cases in the domain of employment law.

In order to deal with open texturedness the approach taken here is to design the knowledge base in such a way as to contain instances, exemplars or paradigms of various categories of past

case. Each new case is analyzed by comparing it to the exemplars that it most closely resembles. This technique is very useful in case-based legal reasoning where exemplars can help fill the gap between case descriptions and the language of generalization.

Generalization by itself cannot be sufficient in legal reasoning. For example generalized features like *sex discrimination* or *equal pay* are clearly too general to retrieve the most genuinely similar cases from the case base. This problem can be tackled by an approach to knowledge representation in which full descriptions of known instances or exemplars are represented and a new case is compared to the exemplars that it most closely resembles. An exemplar-based representation can reason about categories for which there are insufficient generalizations. It requires knowledge of the relations among features and of the explanatory principles that connect exemplars to the categories of which they are members [3].

# 4 Indexing in Multiple Case Multiple Features

A third problem which the current research is investigating involves the problem of indexing. This is a problem in CBR in general, and in legal reasoning in particular. What happens when the indexing mechanism does not retrieve relevant cases even though there is one or more relevant case in the knowledge base? In other words what happens when the index of the target case does not correspond to the one which has been used to try retrieve the past case? A number of approaches have been proposed including index transformation [16], condensation [9], causal explanation and decomposition [11], elaboration [8] and tweaking [15].

When generalized features are used in a large case base many cases are retrieved. Under these circumstances it is hard to decide which one of the cases represents the closest match. To alleviate this problem in our research, we concentrate on higher order features such as *special circumstances.* For example if we have the set of indexes described below;

- *Employment, Sex discrimination, Indirect discrimination, Equal pay, Special circumstances.*

We can use the words *Equal pay* as a generalized feature and *Special circumstances* ( e.g. *red circling* ) as a specific feature. If the specific feature did not match any feature in the source cases and no similar case is retrieved, then some index transformation and elaboration must be done. In the case TRICO FLORETH LTD. V. S. GROVES AND E. AISTON (I.R.L.R. 1976 327 C.M.-117) which is about women who claimed that they should be paid the same wage as men who were doing the same work, the employers resisted the claim on the grounds that the variation in pay was genuinely due to a *material difference* (i.e. other than a difference in sex). These are two very similar cases to the above case and where a different high order index is used.

- SNOXELL AND DAVIS V. VAUXHALL MOTORS LTD. [7]. FEATURES*(Sex discrimination. Equal pay. Material difference. Red circling).*

- CHARLES EARLY AND MARRIOT (WITNEY) LTD. V. SMITH AND BELL (HUNNINGS, 1988 P.186-7). FEATURES*(Sex discrimination. Equal pay. Material difference. Red circling).*

In both of the above cases claims arose under equal pay Act 1970 and the principle issue concerns the correct treatment of the practice known as *red circling* or *red ringing. Red circling* is the practice of protecting the wages of an employee or group of employees, moved from a better paid type of work to a worse paid type of work, perhaps because the first is no longer undertaken. Such transferred employees are often ringed in red in work schedules. It may happen that where men and women undertake like work and where all the women are paid less than any of the men, the discrimination will be justified on the basis that the men are *Red circle* cases [7]. In

the last two cases *red circling* or *red ringing* can be used as a specific index. But *red circling* will not retrieve the first case although it has many features in common. One way to solve this problem is to use an abstract index which it is not discussed in this paper. Another method is to use index transformation and elaboration techniques, by adding more detail to the index. For example instead of using red circling or material difference we can use *better paid type of work to a worse paid type of work*, or *equal pay for men and women for same work.*

## 5 Conclusion

In conclusion we have described preliminary work in the design of a prototype system combining the problems of generalized features, exemplars and indexing in multiple case, multiple features retrieval. Initial results suggest that the choice of employment law as an example domain has yielded an extremely fruitful area on which to test these concepts. Further research will concentrate on building a larger system in which the concepts described above may be further refined and tested. In particular the question of adapting solutions will be investigated.

## 6 References


1. K.D. Ashley: Distinguishing a Reasoning Wedge. In: Proceedings of 9th Annual Conference of the Cognitive Science Society, Lawrence Erlbaum Associates, Hillsdale, N.J. 739-747 (1989)

2. W.M. Bain: Case-based Reasoning : A Computer Model of Subjective Assessment, Ph.D. thesis, Yale University (1986)

3. L.K. Branting: Integrating Generalisations with Examplar-based Reasoning, Proceedings of DARPA Workship on Case-based Reasoning, 224-229 (1989)

4. A.V.D.L. Gardner: An Artificial Intelligence Approach to Legal Reasoning, MIT Press, Cambridge, Massachussetts (1987)

5. K.J. Hammond: Case-based Planning, Ph.D. Thesis, Yale University, reprinted as Case-based Planning, Academic Press : San Diego, CA (1988)

6. H.L.A. Hart: The Concept of Law. Clarendon Press, Oxford (1961)

7. M.N. Hunning.: EEC Employment Cases. European Law Centre (1988)

8. J.L. Kolodner: Retrieval and Organisational Strategies in Conceptual Memory: A Computer Model, Lawrence Erlbaum Associates, Hillsdale, NJ (1984)

9. J.L. Kolodner: Retrieving Events from a Case Memory: A Parallel Implementation, Proceedings of the 1988 Case-based Reasoning Workshop, Clearwater, Fla. 233-249 (1988)

10. P. Koton: Reasoning about Evidence in Causal Explanation: In Proceedings of AAAI-88, American Association for Artificial Intelligence, Morgan Kaufmann publishers Inc., Los Altos, CA. 256-261 (1988)

11. D. Navinchandra: Case-based Reasoning in CYCLOPS, a Design Problem Solver", Proceedings of DARPA Workshop on Case-Based Reasoning, 286-301, May 10-13 (1988)

12. E.L.Rissland K.D. Ashley: Hypotheticals as Heuristic Devices. In: Proceedings of AAAI-86, American Association for Artificial Intelligence, Morgan Kaufmann Publishers Inc. 289-297 (1986)

13. E.L. Rissland, D.B Skalak: CABERET: Rule Interpretation in a Hybrid Architecture, International Journal of Man-machine Studies (1991)

14. K.E. Sanders: Within the Letter of the Law: Reasoning among Multiple Cases, Department of Computer Science, Brown University, Providence, RI, January 17, 1991, Proceedings of DARPA Workshop, 317-326 (1991)

15. R.C. Schank: Explanation Patterns: Understanding Mechanically and Creatively, Lawrence Erlbaum Associates, Hillsdale, NJ (1986)

16. K.P. Sycara: Resolving Adversarial Conflicts: An Approach Integrating Case-based and Analatic Methods, Ph.D. dissertation, School of Information and Computer Science, Georgia Institute of Technology (1987)

# Massively Parallel Case-Based Reasoning with Probabilistic Similarity Metrics

Petri Myllymäki Henry Tirri

University of Helsinki, Department of Computer Science*
P.O.Box 26, SF-00014 University of Helsinki, Finland
email: Petri.Myllymaki@cs.Helsinki.FI, Henry.Tirri@cs.Helsinki.FI

**Abstract**

We propose a probabilistic case-space metric for the case matching and case adaptation tasks. Central to our approach is a probability propagation algorithm adopted from Bayesian reasoning systems, which allows our case-based reasoning system to perform theoretically sound probabilistic reasoning. The same probability propagation mechanism actually offers an uniform solution to both the case matching and case adaptation problems. We also show how the algorithm can be implemented as a connectionist network, where efficient massively parallel case retrieval is an inherent property of the system. We argue that using this kind of an approach, the difficult problem of case indexing can be completely avoided.

## 1 Introduction

In case-based reasoning (CBR) paradigm the dynamic case memory is central to the reasoning process (see the process model in Figure 1) — learning is an inherent part of the process. Although the idea of using a set of representative instances as the basis for a reasoning system is simple in principle, there are many difficult problems related to constructing a case base from data (learning), case matching and case adaptation. Here we do not address the problem of choosing a suitable set of cases, but for our purposes we assume that they are defined by a human expert, or derived from a large database of observations by statistical clustering methods. We will focus on the central problems concerning the reasoning part of the CBR system: case matching and adaptation tasks.

Much of the published work on CBR has concentrated on applying machine learning methods for case indexing, in order to avoid costly comparison of the input with the large set of cases in the case memory during the case matching task [2]. We adopt an alternative approach and show how to construct a massively parallel implementation of CBR confirming to the so called *connectionist* architecture (see e.g. [12]). Connectionist networks are constructed from a large amount of elements with an input fan order of magnitudes larger than in computational elements of conventional architectures. This means that the set of connections of the elements can be used for distortion tolerant storing of large number of cases (represented by high dimensional vectors) by making single elements "sensitive" to a stored item, i.e., to produce a high output for particular subregions in the input space. In our approach the case indexing problem is thus addressed directly at the architectural level where matching can be performed efficiently by using the available parallelism. On the other hand, chip level implementation of massive parallelism constraints the complexity of a single computing element to a limited set of operations and structurally simple local memory. Consequently case-based reasoning, with a knowledge base of high-dimensional cases as the basis for the reasoning process, offers a very natural computational framework for connectionist architectures.

In addition to using connectionist models for avoiding the case indexing problem, we will also propose a uniform solution to the problem of choosing proper metrics for case matching and adaptation. Developing appropriate metrics for case matching and adaptation has in practice lead to heuristic solutions which are hard to justify theoretically. The obvious disadvantages of such approaches are related to the difficulty of interpreting differences in the similarity of the various cases, and to the related problem of discovering the significance of the difference of the attribute values. For example in ad hoc solutions based on uncertainty values, in most cases it is very difficult to interpret if e.g., the difference between values .9 and .99 is less important than the difference between .6 and .8. To avoid resorting to such ad hoc heuristic solutions in

*This research was supported by Technology Development Center (TEKES) and Honkanen Foundation.

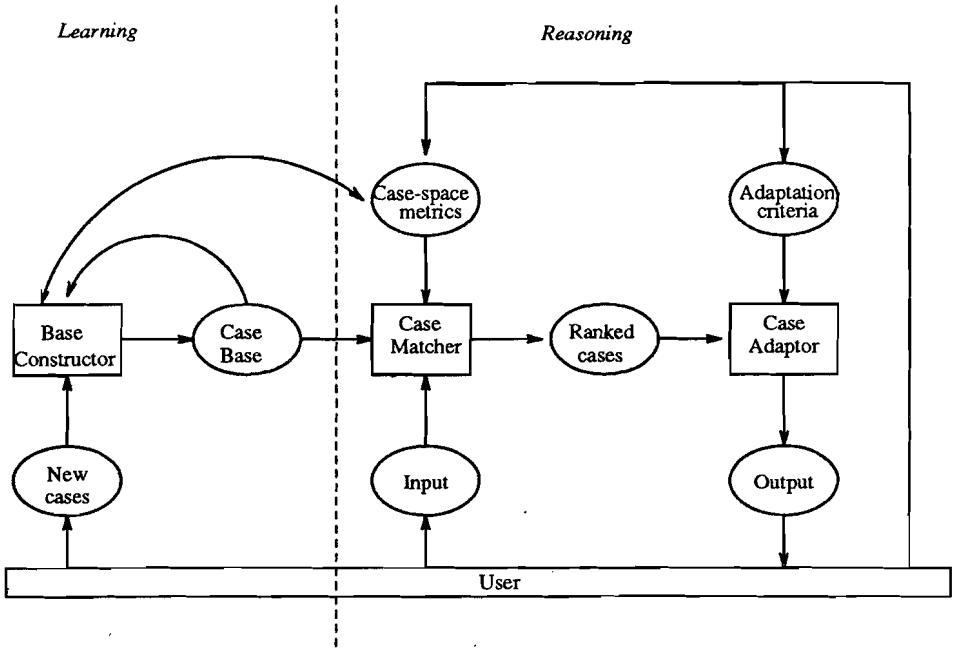



Figure 1: A generic CBR system architecture.

the reasoning process, we propose that one uses methods developed for Bayesian probabilistic reasoning systems [10], [8]. In Section 3, we show how our Bayesian reasoning framework, which we call *Bayesian case-based reasoning*, offers us a uniform similarity metrics for both the case matching and adaptation tasks. In Section 4, we present a massively parallel implementation of our Bayesian CBR system.

## 2 Massively Parallel Case-Based Reasoning

Let our knowledge of the problem domain be coded using $m$ attributes $A_1, \ldots, A_m$. A *case* $C_k$ is represented as a vector $c_k$ consisting of a value assignment for these attributes: $c_k = (a_1(k), \ldots, a_m(k))$, where $a_i(k)$ is either a value of attribute $A_i$, or undefined. Our case base $C$ is a collection of $l$ case vectors, $C = \{C_1, \ldots, C_l\}$.

A CBR process starts when an *input case vector* $c^* = (a_1^*, \ldots, a_m^*)$ is presented to the system. The goal is to provide each case $C_k$ with a *similarity rank* $S(C_k)$ representing the similarity between the vectors $c_k$ and $c^*$ (case matching task), and each attribute $A_i$ not defined in $c^*$ with a value consistent with the highly ranked cases (case adaptation task). For a very general class of case-matching and case adaptation algorithms this can be done in a massively parallel connectionist architecture. As an illustrative example, let us consider a case matching task where the similarity rank for a case $C_k$ is a function depending on the inner product of the case and input vectors: $S(C_k) = F(c_k c^*)$. Let us now assign one processing unit for each of the cases and each of the attributes, and connect each case unit to all the attribute units that belong to the corresponding case value assignment. The weight of the connection from an attribute unit to a case unit is equal to the corresponding attribute value in the case definition. Hence each case vector $c_k$ is coded as a set of weights attached to the connections leading to the corresponding case unit (see Figure 2).

Let the value of an attribute unit be either given in the input case assignment $c^*$, or zero if the value is undefined. In our connectionist network, each unit sends its value to all the adjacent units, which sum all the incoming messages weighted by the connection strengths of the corresponding arcs. It is obvious that the total input of a case unit is the inner product $c_k c^*$. If each case unit now computes the function $F$ using the input as the parameter, we have accomplished our goal: parallel computation of the case ranks. A similar construction can be presented for the case adaptation task. In Section 4 we present a massively parallel implementation of CBR where both the case matching and case adaptation tasks are performed using the same, undirected connectionist network.

As there is nothing that resembles a shared memory, the connectionist computing architecture is inherently parallel, and each element can perform comparison of its input against the value stored in the interconnections independently from the others. This offers a linear speed-up in the comparison

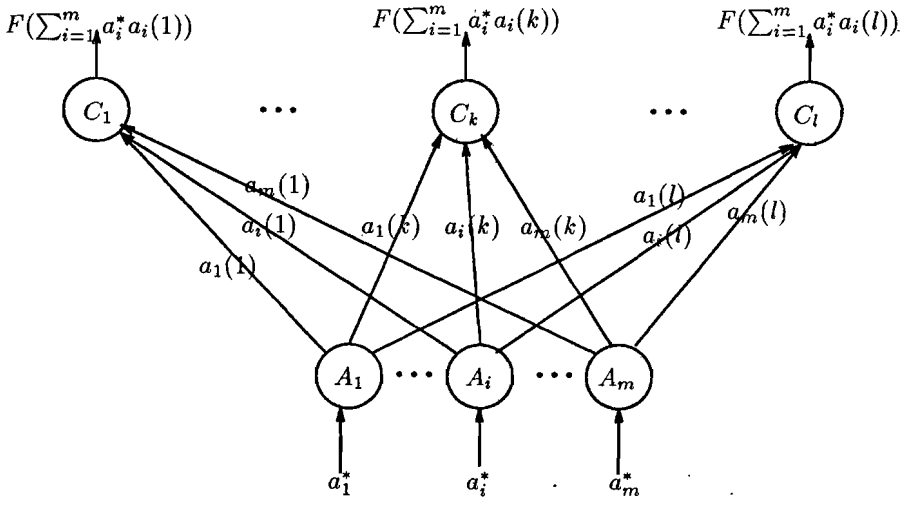



Figure 2: Massively parallel implementation of case matching.

process with respect to the number of computational elements available. This type of a *memory-based* or *instance-based* reasoning approach allows matching of the input against millions of stored cases efficiently [4].

# 3 Bayesian Case-Based Reasoning

In our Bayesian framework, the case attributes $A_1, \ldots, A_m$ are assumed to be discrete random variables. This is a very natural assumption in the context of expert systems, which currently is the main application area of CBR. If an attribute is not discrete, it can be discretized using standard quantization methods. Also all the cases $C_1, \ldots, C_l$ can be regarded as binary random variables, with $C_k = 1$ denoting the fact that case $C_k$ is in question, $C_k = 0$ the opposite situation.

Let an attribute $A_i$ has $n_i$ possible values, $a_{i1}, \ldots, a_{in_i}$. A case vector $c_k$ is a "prototype" representation of a class of (in some sense) similar instances, and is coded as a vector

$$c_k = (\underbrace{P_k(a_{11}), \ldots, P_k(a_{1n_1})}_{P_k(A_1)}, \underbrace{P_k(a_{21}), \ldots, P_k(a_{2n_2})}_{P_k(A_2)}, \ldots, \underbrace{P_k(a_{m1}), \ldots, P_k(a_{mn_m})}_{P_k(A_m)}),$$

where $P_k(A_i)$ expresses the probability distribution for the values of attribute $A_i$ inside the class $C_k$: $P_k(a_{ij}) = P(A_i = a_{ij} \mid C_k = 1)$.

Our case base can now be represented as a Bayesian belief network[10], consisting of variables $A_1, \ldots, A_m$ and $C_1, \ldots, C_l$ (see Figure 3a). Let $C^*$ be a random variable the values of which are the input case vectors, and let $c^*$ denote the current input vector, the value of $C^*$. The theory of graphical belief network representations provides us with rigorous algorithms (see e.g.[10, 8]) for calculating probabilities $P(C_k = 1 \mid C^* = c^*)$ for each case $C_k$ (case matching). What is more, these algorithms offer also a method for computing probabilities $P(A_i = a_{ij} \mid C^* = c^*)$ for all the values $a_{ij}$ not determined by $c^*$ (case adaptation). To be able to use these algorithms, we need to provide each arc from variable $C_k$ to variable $A_i$ with probabilities $P_k(a_{i1}), \ldots, P_k(a_{in_i})$. In addition to this, each case must be provided with a prior probability $P(C_k = 1)$. This probability can be estimated by the proportional number of occurrences of class $C_k$, if a database of observations is available. Similarly, the probabilities $P_k(a_{ij}) = P(A_i = a_{ij} | C_k = 1)$ can be estimated by occurrences of the value $a_{ij}$ inside class $C_k$.

In principle we are now able to solve the case matching task, using the probability measure as the metrics of our system. In addition, the same method can also be used for the case adaptation task of our CBR system. However, the network in Figure 3a is not singly connected, which means that there are loops in the underlying network, if the direction of the arcs is disregarded. In this case, the problem of calculating the above mentioned probabilities can be shown to be NP-hard [1]. One approach to overcome this problem is to use stochastic simulation schemes such as Gibbs sampling [3] for approximating the outcome of the updating process. In our earlier work [9, 6] we presented schemes for implementing Gibbs sampling on a connectionist network architecture. However, the problem of determining the so called annealing schedule has proven very hard in practice, resulting to slow convergence of the algorithm.

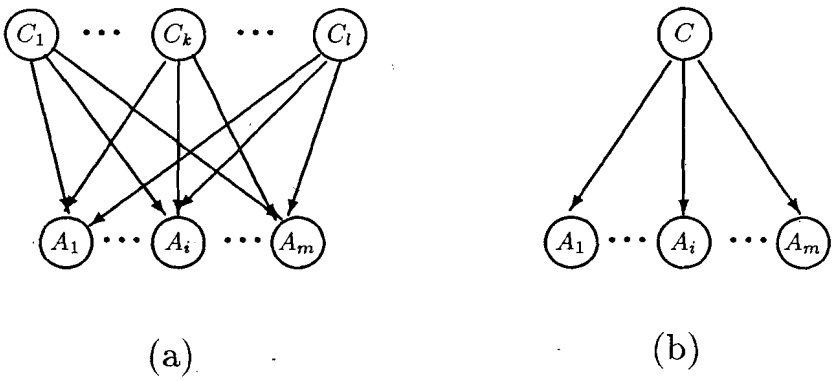



Figure 3: Case base as a a) multiply connected Bayesian network with several case variables, and b) singly connected tree with a single case variable as the root of the tree.

On the other hand, for singly-connected networks, there exists a polynomial time algorithm for belief updating, developed by Pearl [10]. In the approach introduced in [5], a given belief network is first transformed to a singly-connected network, which is then updated by using Pearl's algorithm. However, as the problem is NP-hard, the transformation process may take an exponential time. In the following, we show how our case base can be viewed as a simple singly connected network, a tree, in which case Pearl's belief updating algorithm can be applied directly.

Let us regard the cases $c_1, \ldots, c_n$ as mutually exclusive values of a single random variable, $C$. To be able to do this, all the cases must be *complete*, i.e., all the values $P_k(a_{ij})$ must be given for each case $c_k$. If the user is unable to provide complete cases, the missing probabilities can be filled in by using the uniform probability distribution (if we do not know the value of an attribute, we assume all the values to be equally probable). Alternatively, the user may also define another a priori distribution for the missing cases, if this kind of information about the attributes is available. After storing the complete cases, we can obviously retrieve any probability $P_k(a_{ij})$, given a case $C_k$. In the Bayesian framework this means that *all the variables $A_i$ are conditionally independent of each other, given the value of the variable $C$*. What this means is that the Bayesian network corresponding to this representation is a tree, where a single variable $C$ is the root of the tree, and variables $A_i$ form the leaves (see Figure 3b). To use this network for probabilistic reasoning, an arc to variable $A_i$ must be provided with probabilities $P_k(a_{i1}), \ldots, P_k(a_{in_i})$, for all the cases $C_1, \ldots, C_l$. In the next section, we show how these probabilities can be stored as weights in a connectionist network, and used as part of a massively parallel probabilistic reasoning process.

# 4 Massively Parallel Bayesian CBR

We now show how to construct a undirected 3-layer connectionist network which performs the computations of Pearl's algorithm in parallel. In an earlier paper [7] we discussed a related directed 6-layer feedforward neural network architecture, which has a more complex structure than the connectionistic network presented here, but used simpler computational elements. In addition to the general idea presented in Section 2, we need a special intermediate layer, where for each attribute $X$ we have $l$ nodes, one for each case (see Figure 4). The total number of nodes in the resulting network is $\sum_{i=1}^{m} n_i + ml + l$, and the number of arcs in the network is given by $\sum_{i=1}^{m} ln_i + lm$.

During the network computation process, each node $X$ computes its activation value $S(X)$ using incoming messages, and sends the computed value further through the arcs leading to nodes in the other layers. This activation propagation starts when the user sets the values $S(a_{ij})$ for the nodes in layer 1. According to the idea of *virtual evidence* [10], if there exists some initial evidence $e$ for the value $a_{ij}$ of the attribute $A_i$, the value $S(a_{ij})$ should be set equal to the probability $P(e \mid A_i = a_{ij})$. Total ignorance of the correct value of $A_i$ is represented by setting all the values $S(a_{i1}), \ldots, S(a_{in_i})$ to be equal, for example 1. If the value of $A_i$ is known to be $a_{ih}$ for certain, then $S(a_{ih})$ should be set to 1, and the values $S(a_{ij})$ to 0, for all $j \neq h$.

Intuitively the computation consists of two phases. The initial phase, corresponding to a bottom-up value propagation through the network, performs case matching. As a result, the third layer activation values gives us a matching score for each of the cases (i.e., to nodes $C_k$), thus these activation values can be directly used for classification, if needed. In the second phase, the probabilities are propagated from

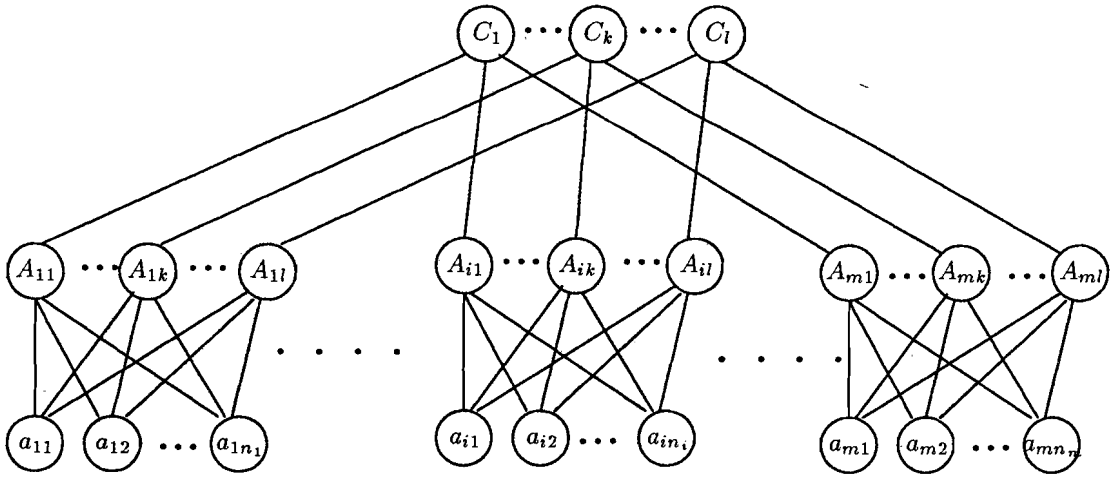



Figure 4: A connectionist network implementing Bayesian CBR.

the top down back to the attribute values, thus complementing the attribute value vectors based on the "winning" cases. In general this allows many stored cases to contribute to the adaptation by the amount justified by their original matching. In the general framework of Figure 1 this approach corresponds to situation where the case-space metrics and adaptation criteria coincide.

In the following we illustrate more closely how the activation propagation process proceeds from layer to layer. The first 3 steps correspond to the bottom-up propagation phase, and steps 4 and 5 to the top-down phase.

**Step 1:** The first layer contains one node for each of the possible attribute values $a_{ij}$, altogether $\sum_{i=1}^{m} n_i$ nodes. The value $S(a_{ij})$ is either given by the user, or initialized to the defined a priori value.

**Step 2:** Layer 2 consists of $m$ groups of nodes, each of which has $l$ individual nodes $A_{i1}, \ldots, A_{il}$, making the total number of nodes in this layer $ml$. Each node $A_{ik}$ has $n_i$ arriving arcs from all the nodes $a_{i1}, \ldots, a_{in_i}$. The weight $W(A_{ik})_j$ from node $a_{ij}$ to node $A_{ik}$ is $P(A_i = a_{ij}|C = c_k)$, i.e., the conditional probability that the attribute $A_{ik}$ has value $a_{ij}$ given an observation from class $C_k$. The activation value of node $A_{ik}$ is computed by $S(A_{ik}) = \sum_{j=1}^{n_i} W(A_{ik})_j S(a_{ij})$.

**Step 3:** In layer 3, there is one node for each of the $l$ classes. Each node $C_k$ receives input from $m$ nodes in the layer 2, $A_{1k}, \ldots, A_{mk}$. The activation value of node $c_k$ is computed by $S(C_k) = \theta_k \prod_{i=1}^{m} S(A_{ik})$. This activation gives a score for each of the stored cases. The constant value $\theta_k = P(C = c_k)$ is assumed to be stored in the node $C_k$.

**Step 4:** The propagation process returns back to layer 2. Each node $A_{ik}$ updates its activation value using the formula $S(A_{ik}) = S(C_k)/S(A_{ik})$.

**Step 5:** Each node $a_{ij}$ on layer 1 receives incoming signals from $l$ nodes, $A_{i1}, \ldots, A_{il}$. The weight $W(a_{ij})_k$ from node $A_{ik}$ to node $a_{ij}$ is $P(a_{ij}|c_k)$. The units update their states by computing $S(a_{ij}) = S(a_{ij}) \sum_{k=1}^{l} W(a_{ij})_k S(A_{ik})$. This can be understood as a "correction" to the original values assuming that the matching cases have prediction value for unidentical, but similar cases.

Using the notation of Pearl in [10], the task of the step 2 is to compute the $m$ values $\lambda_{A_i}(c_k)$, for each of the $l$ cases. As $\lambda(c_k)$ is defined as $\lambda(c_k) = \prod_{i=1}^{m} \lambda_{A_i}(c_k)$, the activation value of node $c_k$ is $S(c_k) = P(c_k)\lambda(c_k)$. Pearl has proved that this is equal to $\alpha P(c_k \mid c^*)$, where $\alpha$ is a normalizing constant. The actual probabilities can now be retrieved easily by normalizing the values $S(c_k)$:

$$P(C = c_k \mid C^* = c^*) = S(C_k) / \sum_{h=1}^{l} S(C_h).$$

In a similar way, step 4 produces the terms $\pi_{A_i}(C_k)$, and step 5 the values

$$S(a_{ij}) = \lambda(a_{ij}) \sum_{k=1}^{l} \pi_{A_i}(C_k) P(A_i = a_{ij} \mid C = c_k) = \lambda(a_{ij})\pi(a_{ij}) = \alpha P(A_i = a_{ij} \mid C^* = c^*),$$

where $\alpha$ is again a (different) normalizing constant. The actual probabilities can be retrieved again by normalization: $P(A_i = a_{ij} \mid C^* = c^*) = S(a_{ij}) / \sum_{h=1}^{n_i} S(a_{ih})$.

Naturally, the two normalization tasks can also be performed in parallel on a connectionist network by using two extra layers of units.

# 5 Conclusion

We have presented an approach where case-based reasoning can be implemented as a connectionist network architecture. The method is based on implementing Pearl's probability propagation as a 3-layer hierarchical network. The advantages of such an approach are twofold. In the first place, it provides an efficient solution to the case indexing problem based on the parallel architecture. Secondly, it offers a theoretically sound Bayesian interpretation of the case-space metrics and its successful application to both the case matching and adaptation via probability propagation. However, it is evident that there are several aspects left for further research. The most important of the questions to be addressed is the proper choice of the cases (observe that in the presence of noise the optimal strategy is not necessarily to store all the cases encountered). In addition, methods to determine the conditional probabilities $P(A_i = a_{ij} | C = c_k)$ used by the reasoning algorithm should be investigated. Initially it can be assumed that such probabilities are estimated by the expert in a regular knowledge acquisition process, but it is clear that one can also use various statistical clustering techniques for this purpose. We are currently developing learning algorithms for our CBR system based on the information theoretic MDL principle [11].

# References


[1] G.F. Cooper, Probabilistic Inference using Belief Networks is NP-hard. Technical Report KSL-87-27, Stanford University, Stanford, California.

[2] DARPA, *Proceedings of the Workshop on Case-Based Reasoning 1988-1990.* Morgan Kaufmann, San Mateo.

[3] S. Geman and D. Geman, Stochastic relaxation, Gibbs distributions, and the Bayesian restoration of images. *IEEE Trans. on Pattern Analysis and Machine Intelligence* 6 (1984), 721–741.

[4] H. Kitano and M. Yasunaga, Wafer Scale Integration for Massively Parallel Memory-Based Reasoning. Pp. 850 – 856 in: *Proc. of the Tenth National Conference on Artificial Intelligence (San Jose, July 1992)* AAAI Press/The MIT Press, Menlo Park, 1992.

[5] S. L. Lauritzen and D. J. Spiegelhalter, Local computations with probabilities on graphical structures and their application to expert systems. *J. Royal Stat. Soc., Ser. B* 1989. Reprinted as pp. 415–448 in: *Readings in Uncertain Reasoning* (G. Shafer and J. Pearl, eds.). Morgan Kaufmann, San Mateo, 1990.

[6] P. Myllymäki and P. Orponen, Programming the Harmonium. Pp. 671–677 in: *Proc. of the International Joint Conf. on Neural Networks (Singapore, Nov. 1991), Vol. 1.* IEEE, New York, NY, 1991.

[7] P. Myllymäki and H. Tirri, Bayesian Case-Based Reasoning with Neural Networks. Pp. 422–427 in: *Proc. of the IEEE International Conf. on Neural Networks (San Francisco, March 1993), Vol. 1.* IEEE, Piscataway, NJ, 1993.

[8] R. Neapolitan, *Probabilistic Reasoning in Expert Systems.* Wiley Interscience, New York, 1990.

[9] P. Orponen, P. Floréen, P. Myllymäki, H. Tirri, A neural implementation of conceptual hierarchies with Bayesian reasoning. Pp. 297–303 in: *Proc. of the International Joint Conf. on Neural Networks (San Diego, CA, June 1990), Vol. I.* IEEE, New York, NY, 1990.

[10] J. Pearl, *Probabilistic Reasoning in Intelligent Systems: Networks of Plausible Inference.* Morgan Kaufmann, San Mateo, CA, 1988.

[11] J. Rissanen, *Stochastic Complexity in Statistical Inquiry.* World Scientific, 1989.

[12] D.E. Rumelhart and J.L.McClelland (Eds.),*Parallel distributed processing: explorations in the microstructures of cognition. Vol 1,2.* The MIT Press, 1986.

# Similarity in Legal Case Based Reasoning as Degree of Matching between Conceptual Graphs: Work in Progress

Jonathan Poole

Computer Science Department, University College London
email: jpoole@uk.ac.ucl.cs

**Abstract**

This project is an examination of the use of structured, semantic network, representations of cases for Case Based Reasoning. The Conceptual Graph notation is used to represent complex events and the aim is to investigate and evaluate methods of defining similarity and adaptation methods for them. The domain for investigation is the Law of Negligence, where the alleged ability of semantic networks to represent causal information is expected to be particularly relevant.

## 1 Introduction

The purpose of this project is to examine the use of structured, semantic network based representations of cases in Case Based Reasoning. In an effort to avoid the creation of *ad hoc* concepts and relations the intention is to use results of previous work on semantic networks and on meaning representations in computational linguistics (in particular Somers' grid of deep cases [10] which is suggested in [2] as particularly appropriate for representing legal cases). As explained in a later section an aim is that the case representations could be parsed from natural language descriptions.

The notation used is that of Conceptual Graphs, though not restricted to Sowa's concepts and relations. The choice of domain for investigation was governed by several principles:

1. it should be complicated enough that the representation and inference rules are thoroughly tested.
2. the cases should not just make sense to a very small set of domain experts. There should be enough common-sense knowledge involved that the problems arising can be demonstrated to all interested parties.
3. a set of well-defined and generally accessible cases should be available to allow other researchers to attempt alternative representations of the same cases.
4. the cases should have enough causal or temporal structure that a structured, semantic network, representation is worthwhile.

The domain chosen is the Law of Negligence (and Nervous Shock), and as far as possible cases chosen are ones generally described and analyzed in legal casebooks. Legal expertise is sought where it seems necessary, but the focus is on dealing with the common-sense level of similarity[1] rather than attempting to simulate the reasoning of legal experts. There is no expectation that the results will be of interest to legal theorists.

An example of a case representation is given in Figure 1 below. The case represented is that of *McLoughlin v. O'Brian [1982]*. In this case the plaintiff claimed (successfully) for the nervous shock caused when she discovered (two hours after the event) that one of her daughters had been killed and other members of her family injured in an accident caused by the negligence of the defendant. Note that this example uses Sowa's case relations rather than Somers' less generally known ones. Many aspects of the representation are arguable, but it gives an idea of the general form and complexity of the representation. Explanatory information and the decision itself are not represented in this example, as the initial aim is to see the requirements of a system that just matches fact descriptions.

[1] an informal example of common-sense (conceptual) similarity might be that 'Mary threw a brick at Bill' is (arguably) more similar to 'Mary punched Bill' than it is to 'Mary threw a tennis ball at Bill', despite the fact that it shares more individual concepts with the latter.

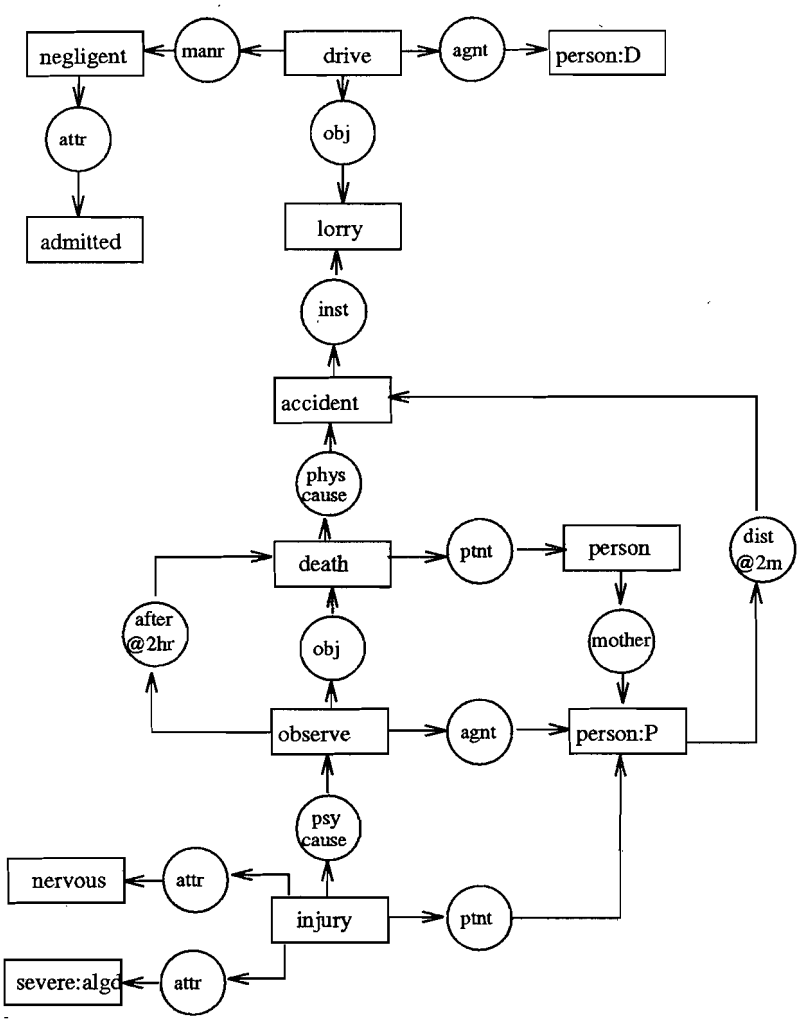



Figure 1: Example Case: McLoughlin v. O'Brian and others 1976

## 2 Similarity and Adaptation with Structured Representations

Structured representations present some particular problems for similarity measurement over and above those for attribute-value representations. In the simplest case, where the Conceptual Graphs of the two cases to be compared are isomorphic, the similarity of the two cases can perhaps be defined as a function of the similarity of the aligned[2] concepts and relations in the two graphs (this is suggested as a similarity metric for CBR in [8]). Even within this very simplistic scheme for comparing structured representations, however, there are complications - particularly with regard to defining a static similarity metric for individual concepts (see below).

More importantly, however, it is hard to guarantee that two Conceptual Graphs with the same meanings[3] are isomorphic. One of the goals of Conceptual Dependency theory (eg [9]) was to define a single, canonical representation of the meaning of any utterance, regardless of the particular words (or even language) chosen. It is not clear whether such an aim is achievable, in fact it seems Schank and his co-workers now use much higher-level representations, that are 'expanded-out' according to need (see review of CD theory [7]). This suggests that even cases that can be said to have identical meaning can potentially be represented in different ways. Clearly cases that are merely similar are going to share much less structure, so strictly isomorphism-based matching is going to have limited use.

In fact it is found with legal cases that even cases that are quoted as direct precedents tend to lead to representations that are very far from isomorphic. How can this problem of lack of isomorphism be tackled? In this research project there are two general approaches adopted:

1. *Graph Grammar rules:*[4] a set of allowable transformations is defined that change the shape of the

[2]Medin *et al* [3] describe *alignment* as the process of deciding which features of one object are to be compared to which features in another object when the two objects are compared

[3]the term *meaning* is used rather informally throughout this abstract. It awaits formal definition.

graph while preserving its meaning. Most of these rules might be 'common-sense' transformations, for example 'if two actions have the same physical cause and happen at the same time they also have the same location'[4]), while some would need to be domain-specific.

2. *Abstraction:* an often used technique to compare semantic networks is to progressively abstract the graphs until a common abstraction is found (eg [6]). Three (related) methods of automatically abstracting graphs are considered:

   - *abstracting individual concepts:* particular concepts (or relations) might be locally replaced by generalizations (ie concepts higher up a concept hierarchy).
   - *attribute dropping:* concepts that can be identified as 'least important' to the meaning of the case may be dropped.
   - *converting subgraphs to nodes:* subgraphs describing a part of the case might be replaced by a single node approximating the meaning of the subgraph. (Clearly this method is very closely related to the use of a graph grammar mentioned above.)

Each of these techniques is potentially complicated, and in general it will not be possible to guarantee that they will not alter the meaning of the case transformed. Thus it is necessary to associate some form of certainty factors with transformations, to ensure cases are not transformed beyond a point at which they might be expected to preserve their meaning.

## 3 Similarity of individual concepts

As part of any similarity metric between semantic networks there will need to be a sub-metric of similarity between individual concepts. Some schemes considered include:

- *Predefined metric:* each concept is assigned a similarity rating with each other (for example in a matrix). If the possible concepts to be represented cover a sizeable proportion of natural language concepts then manually producing such a metric will be impractical.
- *Reduction to primitives:* each concept is reduced down to a subgraph of a limited set of primitive concepts and relations (as in Schanks Conceptual Dependency theory eg [9]). These primitive relations can then more easily be compared to each other. There are many open questions here regarding what primitives might be used and how far this process should go, and it is not clear how helpful primitive representations are.
- *Comparison of features:* if each concept used is defined in terms of a set of features then similarity can be defined in terms of number of shared features, perhaps with weighting of particular features.
- *Traversal distance in a concept hierarchy:* if all concepts are in a hieradaterchy (as in [11]) then the similarity between two concepts can be defined in terms of the number of edges traversed between the two concepts, or the depth in the hierarchy of the least common supertype.

Each of these schemes has separate problems but there is one problem of particular interest that they all share: the problem of the context of the concepts.

## 4 Dealing with Context in Similarity Judgements

In the Psychology literature the subject of similarity is an important one (see [3] for a recent review). Within that literature it is generally held that there can be no fixed, context-independent similarity rating between two concepts: similarity only makes sense 'with respect to' something.

As a practical example, imagine comparing two legal cases, within one case is the concept 'dog', in the other is the concept 'hi-fi'. If these two concepts are aligned, then their similarity needs to be assessed. This could be done by counting edges traversed in a concept hierarchy hierarchy, or counting up shared and non-shared features to arrive at a rating. However, the problem is that the similarity of the concepts depends on their roles in the cases. If the two concepts represent objects that have been stolen, then their monetary value might be seen as the most important feature to be compared. If they have been causing a nuisance their ability to make noise might be important. If they have been intentionally damaged (harmed) then different criteria again might apply.

[4] clearly this rule would actually require more qualification than given here

This problem does not quite arise in this form with attribute-value representations, as the name of the attribute in a sense fixes its context. If there are very specific predicates such as 'EMPLOYEE RECEIVED SOMETHING OF VALUE TO SWITCH EMPLOYMENT', where the values allowed include the list of items received ([1]) then it can be known which features of the items are relevant, and so perhaps it is possible to set up a static metric.

Even with attribute-value representations, however, the subject of context is sometimes seen as relevant. Thus Cain *et al* [12] use 'explanation based learning' techniques to determine how important individual features are in terms of their role in an explanation of the case using domain knowledge. The weight given to individual similarity matches is therefore dependent on the context of the other features in the particular case.

One approach to context considered in this project is within the framework of using a concept hierarchy to determine similarity. All common supertypes of the two concepts matched are considered. The aim is to find the least common supertype that could play the role of both individual concepts in the two graphs.

Thus in the example of the dog and the hi-fi comparison, common supertypes might be 'valuable-object', 'producer-of-noise', 'object-of-size-about-x' and so on. Inferences can be made to test which of these can sensibly fit into the position of the concepts in the separate graphs. Similarity could then be a function of the depth in the concept hierarchy of this least common supertype.

There are other aspects to the problem of context, and part of the research project is to characterize the problem in detail.

## 5 Creating Representations Directly from Natural Language

An important feature of the representation used is that it be (theoretically) possible to create it from natural language by machine. Partly this is because of a desire to use the similarity metric results in areas such as information retrieval, and partly to avoid the creation of *ad hoc* concepts and relations.

At present the issue of *actually* undertaking the translation has not been tackled, but the intention is that the representation be something that a computational linguist would recognize as a representation of the meaning of the original natural language. The aim would be a representation form and set of transformation and matching rules that fulfil Hirsts[5] desiderata for a semantic interpreter of natural language (*ibid.* p. 137) , in particular the idea of *compositionality:* '[...] We would like each syntactically well-formed component of a sentence to correspond to a semantic object, and we would want that object to retain its identity even when it forms part of a larger semantic object.'

## 6 Experimental Evaluation

The intention is to evaluate the ideas described experimentally as well as theoretically. The following is a simple example of the sort of evaluation it is hoped will be able to be carried out. A set of subjects view a series of events (similar to events in legal cases). Each is then asked to describe these in their own words. The descriptions are converted algorithmically into meaning representations. The test of the system would be whether it is able to match all the alternative descriptions of the same events, and distinguish representations of different events.

# References


[1] Kevin D. Ashley. *Modelling Legal Argument: Reasoning with Cases and Hypotheticals.* The MIT Press, 1990.

[2] Judith P. Dick. Representations of legal text for conceptual retrieval. In *Proceedings of the Third International Conference on Artificial Intelligence and Law*, pages 244–253, 1991.

[3] Robert L. Goldstone Douglas L. Medin and Dedre Gentner. Respects for similarity. *Psychological Bulletin*, 100(2):254–278, 1993.

[4] Hartmut Ehrig. Introduction to graph grammars with applications to semantic networkss. *Computers and Mathematics with Applications*, 23:557–572, 6–9 1992. Also available in a collection titled *Semantic Networks in Artificial Intelligence* edited by F. W. Lehmann.

[5] Graeme Hisrt. Semantic interpretation and ambiguity. *Artificial Intelligence*, 34:131–177, 1988.

[6] Robert Levinson. Pattern associativity and the retrieval of semantic networks. *Computers and Mathematics with Applications*, 23:573–600, 6–9 1992. Also available in a collection titled *Semantic Networks in Artificial Intelligence* edited by F. W. Lehmann.

[7] Steven L. Lytinen. Conceptual dependency and its descendants. *Computers and Mathematics with Applications*, 23:51–73, 6–9 1992. Also available in a collection titled *Semantic Networks in Artificial Intelligence* edited by F. W. Lehmann.

[8] Sung H. Myaeng and Aurelio Lopez-Lopez. Conceptual graph matching: a flexible algorithm and experiments. *Journal of Experimental and Theoretical Artificial Intelligence*, 4:107–126, April–June 1992.

[9] R.C. Schank. Identification of conceptualizations underlying natural language. In R.C.Schank and K.M. Colby, editors, *Computer Models of Thought and Language*. Freeman, San Francisco, 1973.

[10] H. L. Somers. *Valency and Case in Computational Linguistics*. Edinburgh University Press, 1987.

[11] John F. Sowa. *Conceptual Structures: Information Processing in Mind and Machine*. Addison-Wesley Pub. Co., 1984.

[12] Michael J. Pazzani Thomas Cain and Glenn Silverstein. Using domain knowledge to influence similarity judgements. In *Proceedings of the DARPA Workshop on Case-Based Reasoning*, pages 191–199, 1991.

# A similarity-assessment algorithm based on comparisons between events

Sophie Rougegrez

LAFORIA-IBP
Université Pierre & Marie Curie
4, place Jussieu
75 230 PARIS cedex 05
FRANCE
e-mail : rougegre@laforia.ibp.fr

**Abstract.** This work is related to the prediction of process behaviours. From the description of a process behaviour since its beginning, the aim of our system is being able to predict its follow-up. In the domain considered, the use of a model is not sufficient. The only reliable knowledge consists in a set of the behaviours of some processes. Their utilization has led to a case-based approach. A case describes the behaviour of a process with sucessions of events. As events may be quite different, cases are considered from a viewpoint. In this paper, we describe how matching is realized thanks to a string matching algorithm.

## 1 Introduction

CBR consists in searching in memory for a problem similar to the problem to resolve and adapting its solution, when necessary, according to the differences between their terms [6]. So that the similarity measure be accurate, a case has thus to represent relevant pieces of information on the problem. This requires a good understanding of the initial episodes [4].

Our system has to predict the behaviour of one process, more precisely its follow-up. The system receives information on the process behaviour, from which it has to predict what will happen afterwards. This problem is not new. Prediction is indeed an important field of AI. But working realized in this area has always concerned domains for which the process could engage in a limited number of behaviours [1]. Such is not the case with regard to the domain chosen for this work, forest fires. In this domain, process developments depend on many interacting parameters that may take an infinity of values.

There exists some models that can help to predict the propagation of a fire, i.e. to know where the fire will be and when. Some of them evaluate propagation speed. These models don't take explicitely into account parameters that the experts consider as the most important for a fire development : relief, vegetation, wind. Actually, existing models restrict a fire to its combustibility properties.

Little knowledge is available about the influence of these last parameters on a process behaviour. These models take into account the influence of the value of one parameter at one instant. The approach chosen is different, we consider indeed the history of parameters : if during their development, parameters take the same succession of values, then two processes are likely to behave the same way.

This is the reason why we use case-based reasoning. While in CBR some features are extracted from the initial description of experiences and make up the representation of a case, a case describes here the totality of a process behaviour, including the values of the parameters that influenced it.

Processes result from many different types of parameters. Despite of this, cases have been represented in a uniform formalism, the one of "event". But such a representation does not permit to index the case base. Selection and matching phases are thus mixed : the algorithms are described in this paper.

## 2 The Domain : an Overview

A forest fire propagates under the influence of many parameters such as wind, relief, vegetation [7]. It can run across different accidents of relief, different vegetations. And during a fire development, several winds with different directions and speeds may coexist. That's the combination of these different parameters that determines the propagation.

To fight efficiently against fires, firemen have to anticipate the propagation. It is realized in the field thanks to the observation of some parameters like those above. Some data can be calculated. Propagation speed for example is very critical. Its prediction is realized thanks to some models of propagation [2].

They rely on a combination of parameters too. But they consider the value of some parameters at one instant only, whereas the influence of parameters like relief, vegetation and wind is not limited to one instant but to the whole of the fire follow-up. The clearing of a valley may for example accelerate it or change its direction. We consider then that if during their developments, two fires go through the same values of parameters, e.g. run along the same relief, then they will behave the same way afterwards. Some of the parameters are quite stable, we concentrate here only on changing ones.

## 3 Method being used

This method which relies on similarities between histories of parameters values is case-based and runs the following way : from the description of a process since its beginning, i.e. a target case, the system has to search for a process, a source case, which is gone through the same succession of values of parameters. Then it uses the behaviour that followed to predict the one of the target case [5].

The searching of an history of parameters requires a partial order between values of parameters. We introduced two types of parameter order : one relating to time and the other relating to physical distances. They are non-conflicting.

### 3.1 Initial Description of Fires : some Properties

The reports written by firemen describe the environment in which the fire occured, its propagation, and the value of some parameters : the relief and the vegetation run along, the changes of wind. In the follow-up we consider that these parameters take successions of values, extracted directly from the reports or resulting from a transformation. One of them is described in section 4.

As we have to locate an history of fire development in an other one, we need to introduce a partial order between values of parameters making them up. This is realized in the reports thanks to the fire departure location and date. Relief is initially described thanks to a curve. The points that make it up are located a certain distance away from the fire start. It is the same for vegetation. Wind is located in space and in time but it is mainly time that permits to order its successive values.

We assert the following :
Let v1 and v2 be two values of parameters, either relief, vegetation and wind and i1, i2 be the instants/distances associated with them then we have :

| v1 occurs before v2 iff i1 before i2, if i1 and i2 are dates<br>iff i1 < i2, if i1 and i2 are distances | (*) |
|---|---|

Relation (*) can be false in two cases :
- if v1 and v2 are located thanks to some distances and fire spreads backwards. But this is quite seldom and in our system we neglected this case,
- a fire can spread in several directions and speeds may be different in each one. We alleviated this problem by considering and representing separetely different directions of propagation.

Relief has a continuous description, whereas vegetation representation is segmented and wind description is discrete. Despite of this, we utilize a uniform representation in terms of events.

### 3.2 A Uniform Representation for Heterogeneous Parameters

Matching between two cases needs to take into account the succession of values of all the parameters considered. But inside each case, these parameters may be quite different and we can consider that we will never find a source case matching a target case according to all of them. Indeed, even if two fires happen at the same place, experts recognize that fire behaviours may be very different : only the relief may not change but all the other parameters would. That's the reason why we decided to match cases from a viewpoint. Matching from a viewpoint consists in comparing cases restricted to one of the parameters influencing propagation [5]. In our system, two cases are then similar from at least one viewpoint.

CBR may be different from one viewpoint to the other one, it is the same for representation of parameters. Indeed, parameters have either a continuous or a discrete description. But each value can be considered as having an effect on the environment on which it has an influence. For that reason, we associate an event with each change of value of one parameter. A change of wind for example constitutes an event.

But which value to choose if a parameter has a continuous description ? This problem arises for relief. In fact, our continous description is tranformed into a discrete one through an association of data.

## 4 From a Continuous to a Discrete Tranformation from the Relief Viewpoint

Matching's aim is to find a relief which exactly matches an other one. But this can seldom occur. And a superposition of curves, initial representation of relief, is not adapted : there can be a difference of height of the points compared, or points at the same height may be shifted and despite this little difference, matching may fail. What seems important for experts is the succession of slopes, of accidents of relief run along. We have at our disposal eight types of forms : valleys, cols, cliffs, etc. And before matching the system proceeds to a transformation of curves into successions of slopes, either descendant or ascendant. Then it tries to associate successive slopes to constitute some forms such as the one mentioned above. This step is realized thanks to a set of rules. The generated forms or remaining slopes make up events. At the end of the transformation, we obtain sequences of events from the viewpoint relief whose an exemplar is given below :

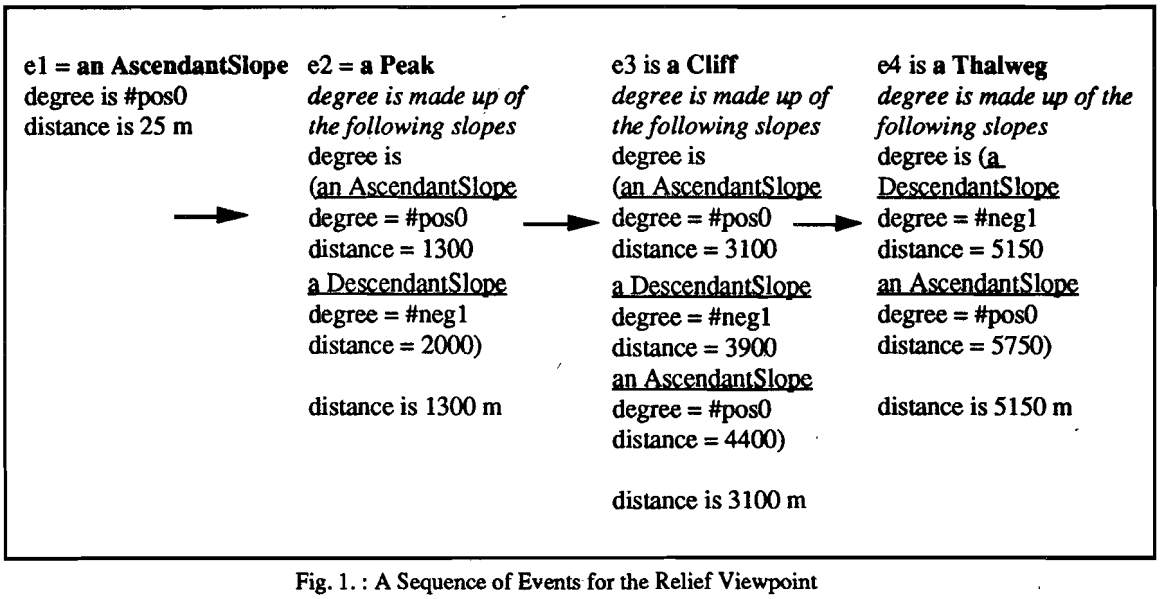



Fig. 1. : A Sequence of Events for the Relief Viewpoint

The degree associated with a form expresses the degree of membership of the form to the model of this form. For forms like peaks, the degree is made up of the slopes that make it up. The distance is a distance of the relief accident from the fire departure location and permits to locate these events on a "distance" axis.

From the other viewpoints, a transformation can be realized. Such is the case of the wind viewpoint for example. It only consists in associating relating events with a propagation axis.

## 5 Case Representation

Whatever the viewpoint considered, all the events are related to the next. That means that events located thanks to some distances (relief, vegetation), are related to the next on the "distance" axis, while those described with time are related to the following on a "time" axis.

Each event conveys an effect on the propagation but our system does not yet consider their combinations. We get then the following case description (fig. 2).

A case describes an achieved propagation of fires in terms of events (represented by circles). A fire can spread in several directions. Only one is represented here. Wind and relief are transformed for matching. Result of transformation is kept for storage in the case base.

Like wind, propagation events are associated with time. Propagation events indicate the progression of fire by a location reached and a time. Hypotheses on future propagation are expressed thanks to these events. The mechanism permitting to generate them is described in [5].

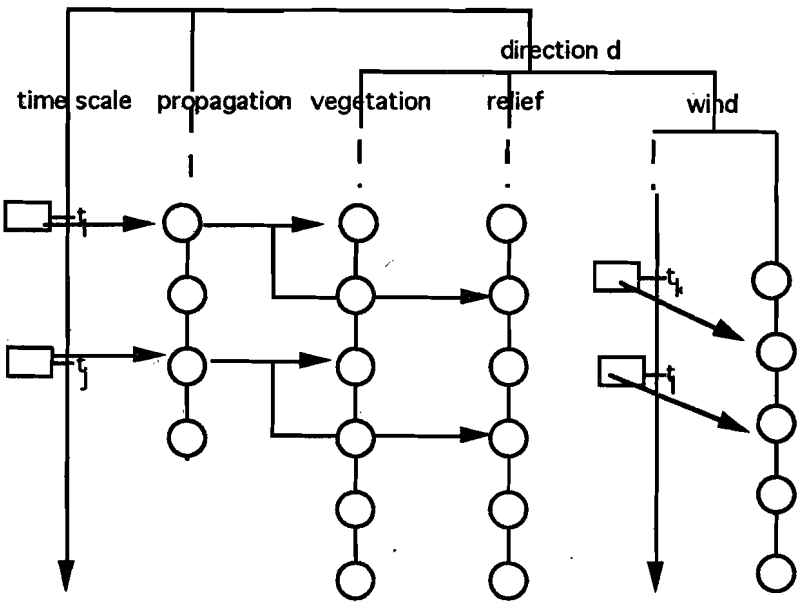



Fig. 2. : The case Representation

## 6 Matching between successions of events

The succession of events to compare is a collection of events relative to the same viewpoint, i.e. relief, vegetation or wind. In our system, matching and selection are mixed.
The following algorithm permits to select the best case :

```
The target case is the case, i.e. fire, whose we wish to predict the future evolution
l _ empty list.
Starget _ the sequence of events relative to the viewpoint v which occured since
the beginning of the target case,
For each case c in the case base do
 for each direction of propagation in c do
   Ssource _ the sequence of events relative to the viewpoint v which occured
   since the beginning of the source case.
   cost _ the littlest cost of matching between Starget and each subsequence of
   Ssource of length the one of Starget.
   add cost to l with the description of the location in the case base of the
   sequence of events relating to it
 end for
end for
best result is the element of l whose cost is the littlest.
```

Fig. 3. : Best case Selection Algorithm from one viewpoint

As told above, matching may be different from a viewpoint to the other one. In the following we limit the description of matching algorithms to one of them : that relative to relief.

Matching between two strings of events considers strings of the same length, i.e. made up of the same number of events. We would like the same events happen in both, and in the same order. But as already told above, this can seldom occur. We have to evaluate a distance between them.

Two strings of events may be different because of
- event content : events may indeed be relative to a valley, a col, etc.,
- sequencing of events : we may have to face the following configurations of strings (fig. 4).

Let A be an event from source case, B, B', C' other events from target case.

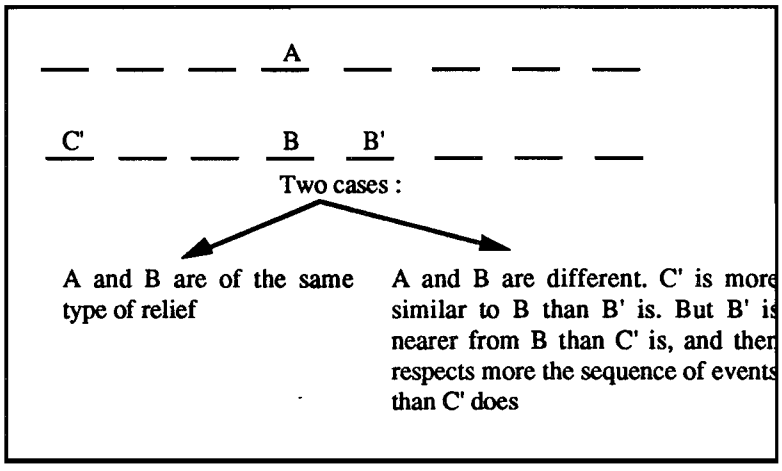



Fig. 4 : An Example of Configuration of Events in two strings

To evaluate similarity between 2 strings of events, we introduced one evaluation of similarity, one relating to the content of events, the other one to their positions.

These measures are evaluated thanks to 2 algorithms. One of them calculates a physical distance between events. Its aim is to match an event of the source case with the nearest event, if possible with the same content, of the target case. It is described in fig. 5.

```
ltarget _ symbol list B describing the relief stemming from the target case.
lsource _ symbol list A describing the relief stemming from the source case of length
the one of B.
distance _ 0. i _ 1.
while ltarget and lsource are not empty do
  ftarget _ ltarget[i].
  fsource _ lsource[i].
  if ftarget and fsource describe the same kind of slope (ascendant or descendant) or the
  same kind of form then
    cost _ comparison between ftarget and fsource
    suppress ltarget[i]
  else
    search for a kind of slope or the kind of form the most like fsource and such that it
    is in ltarget. The found object is aux. We take it off from ltarget.
    cost _ the distance, in number of positions, from the aux position in lsource to
    the ftarget position in ltarget
    i _ i + 1
 end if
 distance _ distance + cost
end while
result _ distance.
```

Fig. 5 : Matching Algorithm Relating to the Position of Events

Last one (fig 6) considers only contents of events and considers both strings as sets :

Both precedent algorithms utilize a similarity between types of events. The first, when it can not find at a position i in both strings the same type of event, searches for another type, the most similar to the first, which may be located at a position near from i.

The second algorithm holds the same operation when it can not find in the target case an event of a given type. Both utilize a distance between types of events based on their structure. It associates with each form the type(s) of form(s) similar to it and the numerical distance to each one.

```
ltarget _ symbol list B describing the relief stemming from the target case.
lsource _ symbol list A describing the relief stemming from the source case of length the one of B.
cpt _ 0. i _ 1.
while i <= length (lsource) do
   cpt _ cpt + the degree of membership of the ith element of lsource to ltarget
   i _ i + 1
end while
result _ cpt.
```

Fig. 6. : Matching Algorithm Relating to the Content of Events

## 7 Related work and Conclusion

This string matching problem is a difficult one. It has to be approximate to reject no solution. Other approximate algorithms exist [3] but they can only match together characters located at the same location or shifted to the right or left of one position. And they tolerate only identical characters.

Here, we consider that events in both strings may be organized randomly. We consider too that events may be different. We introduced then two similarity metrics, one relying mainly on the positions of events compared and the other one on the similarity of their content. Resulting values make up a distance, resulting from the adding of both, which permits to choose the best case.

This type of matching between 2 cases facilitates representation of cases. It doesn't require an interpretation of initial episodes, and relies little on domain knowledge. But it has a drawback which is the lack of precision of similarity evaluation. It can not be used for the adaptation of solution for example.

We are working now on the testing of this algorithm for wind viewpoint. String matching algorithm principle remains the same but distances between events do not. One of the remaining tasks then will be the choice of the viewpoints from which to realize predictions.

## References


[1] : BELEGRINOS P., GEORGEFF M. :
*A model of Events and Processes*
Proceedings of the eleventh IJCAI, vol. 1, 1991.

[2] CASSAGNE H. :
*Modélisation de la propagation des feux de forêts*
Rapport du CEA Cadarache - IPSN/DRS/SESRU, 1991

[3] HALL P. A.F. , DOWLING G.R. :
*Approximate string matching*
Computing surveys, vol. 12, n° 4, pp. 381-402, december 1980

[4] KOLODNER J.
*Improving human decision making through case-based decision aiding*
AI magazine, summer 1991

[5] ROUGEGREZ S.
*A case-based reasoning system independent of a representation of cases in terms of features*
in Proceedings of the AAAI spring symposium on Case-based reasoning and information retrieval,
pp. 98-104, Palo Alto, California, 1993

[6] SLADE S. :
*Case-based reasoning : a research paradigm*
AI magazine, spring 1991

[7] TRABAUD L.
*Les feux de forêts : mécanismes, comportement et environnement*
France sélection, 1992

# A Rule-Based Similarity Measure

Michèle Sebag, Marc Schoenauer

LMS & CMAPX, Ecole Polytechnique
F-91128 PALAISEAU

**Abstract.** A similarity measure based on inductive learning is presented. It requires a set of examples, and some induction algorithm that performs the building of a ruleset from a set of examples ; it is then *knowledge extensive*, compared to approaches requiring the expert to correct the system output and explain his decisions, or to provide and tune numerical coefficients. Several similarity measures can be derived from a ruleset, with a local behavior quite different to that of a weight-based similarity.
A similarity measure enables several tasks beyond the reach of a ruleset, such as clustering the examples or detecting atypical examples. It enables classification as well, by means of a $K$-nearest neighbours method.

## 1 Introduction

A hot research topic in artificial intelligence is about similarity measures, be they concerned with case-based reasoning (CBR) [8, 1], classification [7, 5], generalization in first-order predicate logic [2], or analogical reasoning [11].

In most cases, building a similarity measure requires much knowledge :
• *Declarative* knowledge, as in Protos [1]. The expert, acting as an oracle and a teacher, corrects the system output and explains his decisions ; the various similarity indices involved in Protos evolve through these interactions.
• *Numerical* knowledge, as in KBG [2]. In KBG, predicates are weighted by the expert ; the similarity measure derived from these weights is used to guide the generalization algorithm.
• *Probabilistic* knowledge. In [5], the similarity measure relies on the joined distributions of all variables and all predicates involved in the domain representation. This similarity measure enables to classify examples with incomplete description.

In this paper, a similarity measure only requiring "poor" knowledge, i.e. examples, is presented. Our approach is based on inductive learning : given a set of examples, an induction algorithm is used to build a ruleset [9, 6, 2, 12]. Several similarity measures can then be derived from a ruleset, with a local behavior quite different to that of a weight-based similarity measure.

Section 2 defines the rule-based similarity measures and studies the requirements of our approach. Section 3 presents an experimental validation on three problems well-studied by the machine learning community [9, 4, 3, 12, 7].

## 2 Principle

In machine learning, a rule is traditionnally considered with respect to its extension, i.e. the given examples satisfying the premises of the rule [9]. Reciprocally,

given a set of rules, an example can be associated to the rules whose extensions it belongs to. The subsets of rules associated to examples can then be compared.

## 2.1 Defining rule-based similarity measures (RBS)

Let $Th = \{R_1, \ldots, R_N\}$ ($Th$ like *Theory*) be a set of rules defined on problem domain $E$ ; we associate to any example $e$ in $E$ a subset of $Th$ (maybe empty) : the set of rules $R_i$ whose premises are satisfied by $e$.

$$E \rightarrow \mathcal{P}(Th)$$
$$e \in E \rightarrow e' = \{R_i \in Th \ / \ e \ satisfies \ the \ premises \ of \ R_i\}$$

Given two examples $e_1$ and $e_2$, one can then compare their images $e'_1$ and $e'_2$. The dissimilarity $D(e_1, e_2)$ is set to the cardinality of the symetric difference $e'_1 \Delta e'_2$; two examples matching the same rules thus have zero dissimilarity. Dissimilarity $D$ would then be coarse if sets $e'_1$ or $e'_2$ were trivial, i.e. empty or reduced to a single element. This restriction will be discussed in 2.3.

We accordingly define the rule-based similarity (*RBS*) of two examples:

*Definition 1. Similarity $S_1$ is a function defined from $E \times E$, where $E$ denotes the problem space, onto $R^+$ : $S_1 : E \times E \rightarrow R^+$*

$$S_1(e_1, e_2) = \#\ \{R_i \in Th \ / \ (\ e_1 \ and \ e_2 \ satisfy \ the \ premises \ of \ R_i) \ OR \ (\ neither \ e_1 \ nor \ e_2 \ satisfy \ the \ premises \ of \ R_i)\ \}$$

*where '#G' stands for the number of elements in set G.*

It may seem a bit artificial that rules fired by none of two examples contribute to their similarity. So a second *RBS*, named $S_2$, only takes into account rules actually fired by both examples.

*Definition 2. Similarity $S_2$ is a function defined from $E \times E$ onto $R^+$, by :*

$$S_2(e_1, e_2) = \#\ \{R_i \in Th \ / \ (\ e_1 \ and \ e_2 \ satisfy \ the \ premises \ of \ R_i)\}$$

Last, the relevance of rules can be taken into account by means of weights :

*Definition 3. Define the weight of rule $R_i$ as*

$$w(R_i) = w_i = \#\{positive \ examples \ of \ R_i \ in \ the \ example \ set\}$$

*Then, similarity $S_3$ is a function defined from $E \times E$ onto $R^+$, by :*

$$S_3(e_1, e_2) = \sum_{(\ e_1 \ and \ e_2 \ satisfy \ the \ premises \ of \ R_i)} w_i$$

**Remark** : All definitions above are operational whatever formalism examples and rules are expressed within : they only need to check whether or not an example satisfies a rule.

## 2.2 Comparizon with a weight-based similarity

In attribute-value formalisms, dissimilarity measures usually rely on weights :

$D_V(e_1, e_2) = \sum_{i=1}^{N} v_i d_i(e_1, e_2)$, or $D_V(e_1, e_2) = (\sum_{i=1}^{N} (v_i d_i(e_1, e_2))^2)^{1/2}$

where weight $v_i$ reflects the relevance of attribute $i$ and $d_i(e_1, e_2)$ denotes the difference between values of the $i$-th attribute for examples $e_1$ and $e_2$.

The difference between a rule-based dissimilarity and a weight-based dissimilarity is illustrated in the context of *"Green pea recognition"* by examples below:

| | *Color* | *Shape* | *Size* |
|---|---|---|---|
| $e_1$ | Green | Circle | Small |
| $e_2$ | Blue | Circle | Small |

| | *Color* | *Shape* | *Size* |
|---|---|---|---|
| $e_3$ | Green | Triangle | Medium |
| $e_4$ | Blue | Triangle | Medium |

A weight-based similarity (*WBS*) should give the same similarity between $e_1$ and $e_2$, and between $e_3$ and $e_4$ : their respective differences are the same (the former is *green* while the latter is *blue*), so the difference estimation only depends on the distance between *Green* and *Blue*, and on the weight of attribute *Color*.

In opposition, let us consider the *RBS* defined from the unique rule

*R : If (Color = Green) and (Shape = Circle), Then Green_Pea*

Rule $R$ makes a difference between $e_1$ and $e_2$ (because it is matched by $e_1$ and not by $e_2$ ; so $D(e_1, e_2) = 1$), but it does not make any difference between $e_3$ and $e_4$ (neither $e_3$ nor $e_4$ does match rule $R$ ; $D(e_3, e_4) = 0$).

A *RBS* enables to make a difference among differences, such as between the pairs $(e_1, e_2)$, and $(e_3, e_4)$. Therefore we claim that the topology induced by a *RBS* may be very fine ; a difference between values of attribute $i$ (instead of being considered always with a given weight) may be either unseen or very influent, depending on the values taken by the examples for this attribute and for the others - and according to ruleset $Th$.

### 2.3 Requisites

Let us now consider the defects of a ruleset as characterized in [10] and study their impact on the *RBS*.

**Redundancy.** Roughly speaking, the redundancy of a ruleset is the average number of rules (leading to the same conclusions) fired by an example. Redundancy is often considered a defect in a rule-based system : it endangers the consistent evolution of the system. Now let us consider a non-redundant ruleset ; assume that any example fires a single rule. Then, any two examples either fire the same rule - and they are similar, or they do not, and they are dissimilar. In other words, a non-redundant ruleset induces a coarse dissimilarity on the problem space. So, the redundancy of the ruleset is mandatory in order to induce a usable dissimilarity.

**Incompleteness.** The incompleteness of a ruleset is manifest as some examples do not fire any rule. The corresponding rule-based dissimilarity does not allow to separate such examples ; so this defect is quite penalizing from our point of view.

**Inconsistency / Errors.** A ruleset is inconsistent when rules leading to incompatible conclusions (e.g. distinct diagnosis) are fired by one example. A rule is erroneous if there exists examples satisfying the premises of the rules, but not its conclusion. These defects are unseen in our approach, as the conclusions of the rules are never taken into account.

In short, the central requisite of our approach is the redundancy of the ruleset.

## 3 Application to Classification

Some experimental validation of our approach is done on 3 well-studied classification problems. The rule-based similarity is first compared with a classical weight-based similarity and a weight-based similarity optimized through genetic

algorithms [7]. The predictive accuracy of the rule-based similarity is also compared to those of the very rules it is based on, and with rulesets induced by some famous induction algorithms [9, 4, 3, 12].

## 3.1 Experimental parameters

We followed the protocol used in the reference litterature. The data set is divided into a training set and a test set; this selection is done at random, except that the classes distributions in the training set are same as in the total data set. Rules are learned from the training set. Validation is done through a cross validation technique ; the results obtained on the test set are averaged over five independant selections of the training and test sets.

We used a star-like induction algorithm detailed in [12] ; similarities $S_1, S_2$ and $S_3$ (2.1) are derived from the rulesets learned from the training set. Any similarity *plus* the training set enables a $K$-nearest neighbours method , denoted RKNN (for *Rules based K-Nearest Neighbours*). The sensitivity of classifier RKNN is studied with respect to the rules redundancy - which is tunable in our generalization algorithm. The redundancy rate ranks from 1 (concise rules) to 5 (the total number of rules is multiplied by about 2.5).

## 3.2 Comparizon with similarity-based classifiers

Two problems fitting within attributes-values formalism are considered. The first one (*Iris*) is the iris data set of Fisher, with 150 examples divided into 3 classes and described by 4 attributes. The second one (*Glass*) is composed of 214 examples divided into 6 classes and described by 9 attributes.
The reference results of J. Kelly and L. Davis [7] are given in *Table 2* ; *KNN* denotes a classical $K$-nearest neighbours method using a weight-based similarity with equal weights. *GA-WKNN* denotes a $K$-nearest neighbours method using a weight-based similarity whose weights are optimized by genetic algorithms[1]. Our results are given in the *RKNN* column, with a redundancy rate ranking from 1 to 5.

*Table 2 : Comparizon with weight-based similarities*

| | *KNN* | *GA-WKNN* | *RKNN* | | | | | | | | |
|---|---|---|---|---|---|---|---|---|---|---|---|
| | | | red. 1 | | | red. 3 | | | red. 5 | | |
| | | | $S_1$ | $S_2$ | $S_3$ | $S_1$ | $S_2$ | $S_3$ | $S_1$ | $S_2$ | $S_3$ |
| **IRIS** | 90 | 94 - 93 | 92 | 91 | 91 | 93 | 93 | 93 | 91 | 91 | 91 |
| **GLASS** | 58 | 60 - 62 | 65 | 64 | 64 | 64 | 68 | 68 | 52 | 70 | 70 |

## 3.3 Comparizon with rule-based classifiers

The comparizon with some well-known induction algorithms is done on a medical problem still fitting within an attribute-value formalism. The data set is composed of 286 examples described by 13 attributes and divided into 2 classes.

[1]The results found in [7] are labelled *Classification Error Rates*. To ease the comparizon we take the complement to 100 % of these results as *Classification Success Rates*.

Table 3 shows the results obtained by *AQ15* [9], *CN2* [4], *Assistant86* [3] and a simple bayesian classifier denoted *Bayes* ; these results can be found in [3]. On our side, the results obtained from just the rulesets are given with the legend *RO* (for *Rules Only*) ; the results of RKNN using the different similarities are given beside, with the redundancy rate ranking from 1 to 5.

*Table 3 : Comparizon with classical induction*

| | Training set (190 examples) | Test set (87 examples) | | | | |
|---|---|---|---|---|---|---|
| *Bayes* | 97 | 65 | | | | |
| *AQ*15 | 100 | 72 | | | | |
| *Assistant*86 | 92 - 95 | 62 - 68 | | | | |
| *CN*2 | 76 - 72 | 70 - 71 | | | | |
| *RKNN* | | | | | | |
| | *RO* | *RO* | $S_1$ | $S_2$ | $S_3$ | red. |
| | 87 | 73 | 71 | 72 | 72 | 1 |
| | 90 | 72 | 71 | 72 | 72 | 3 |
| | 90 | 70 | 71 | 71 | 73 | 5 |

### 3.4 Discussion

The results above are interpreted as follows :

In the *Iris* problem, all classes are equally represented, and all classifiers nearly reach the same results (in particular, all similarities give the same results) ; so, our only advantage compared to the weight-based similarity (with weights optimised by genetic algorithms) is to be less expensive. (about 10 minutes on a Symbolics Ivory-based Lisp machine, against 10 seconds on a HP 700 work station).

On the ill-distributed problem *Glass*, similarity $S_1$ outperforms similarity $S_2$ and $S_3$ when redundancy is low ; the inverse is true when redundancy is high. This can be explained as follows. When redundancy increases, a lot of rules are fired by none of any two examples ; these examples are thus similar for $S_1$ ; in the meanwhile, $S_2$ and $S_3$ improve as expected and our results are significantly better than the reference results (from 5 to 8 points).

On the third problem there is no great difference between all similarities. The predictive accuracy of the rule-based similarities is quite similar to that of the rules themselves (and equal to the best reference results). However, the accuracy of the *RBS* increases as the redundancy increases, while the accuracy of the ruleset decreases. This suggests that the rule-based similarity could be used in order to break the ties.

## 4 Conclusion and Perspectives

This paper describes the 2-step induction of similarity measures given a set of examples. A set of rules is first induced from the examples ; then several similarity measures can be derived from a ruleset.

Our approach appears to put very few requisites on the initial knowledge and requires very few help from the expert compared to most related works (see [1] or [2] ; unfortunately a detailed discussion is prohibited due to space limitations). Besides, it escapes any dependency from the domain representation : the induction

step, if possible, captures the semantic information hidden in the examples whatever their syntactic description. Last, it applies within any formalism ; its only requirements are an induction algorithm to be available within this formalism, and this algorithm to provide redundant rulesets.

Our approach is validated on some well-studied problems, with predictive accuracy equal or slightly better than reference results. Besides, it is worth mentioning that a similarity measure may be used to many other aims than a ruleset : it enables using data analysis tools to pre-process the data, so to detect and discard atypical examples before classification, or to cluster the examples in order to reduce a concept formation task to several conjunctive concepts formation tasks.

Further research deals with pruning a set of reference examples, in order to retain only most prototypical examples. Such pruning is expected to both speed up and improve classification.

## Acknowledgements

Thanks to Y. Kodratoff from LRI, Paris-XI Orsay for many discussions and helpful comments.

## References

[1] Bareiss R., *Exemplar-based knowledge acquisition.* Boston, MA, Academic Press.

[2] G. Bisson, *KBG A Knowledge Based Generalizer.* $7^{th}$ International Conference on Machine Learning, R. Porter B. Mooney Eds, Morgan Kaufmann, Austin Texas 1990.

[3] B. Cestnik, I. Bratko, I. Kononenko, *ASSISTANT 86: A knowledge elicitation tool for sophisticated users.* Progress in machine learning, Proc. EWSL 1987, I. Bratko N. Lavrac Eds, Sigma Press

[4] P. Clark T. Niblett *Induction in noisy domains.* Progress in machine learning, Proc. EWSL 1987, I. Bratko N. Lavrac Eds, Sigma Press.

[5] F. Esposito, D. Malerba, G. Semeraro, *Classification of Incomplete Structural Descriptions*, Symbolic-Numeric Data Analysis and Learning, E. Diday Y. Lechevallier Eds, Ed Nova, 1990, pp 469-482.

[6] J.G. Ganascia, *AGAPE et CHARADE, deux techniques d'apprentissage symbolique appliquées à la construction de bases de connaissances*, Thèse d'Etat, 1987, Orsay.

[7] J. D. Kelly, L. Davis, *A Hybrid Genetic Algorithm for Classification*, Proc. IJCAI 1991, J. Mylopoulos & R. Reiter Eds, Morgan Kaufmann Publishers, pp 645-650.

[8] J.L. Kolodner, *Retrieval and Organizational Strategies in Conceptual Memory: A Computer Model.* Lawrence Erlbaum Associates, Hillsdale, NJ, 1984.

[9] Michalski R.S. I. Mozetic, J. Hong, N. Lavrac, *The AQ15 inductive learning system: an overview and experiment.* Proceedings of IMAL, 1986 Orsay.

[10] S. Rajamoney R. DeJong, *The classification, detection and handling of imperfect theory problems.* Proc. IJCAI 1987.

[11] S. J. Russell *The Use of Knowledge in Analogy and Induction* Pitman, London, 1989.

[12] M. Sebag M. Schoenauer, *Incremental Learning of Rules and Meta-Rules.* $7^{th}$ ICML, R. Porter B. Mooney Eds, Morgan Kaufmann, Austin Texas 1990.

# Case-Based Information Retrieval

Malika Smaïl
e-mail : malika@loria.fr
Tel : (033) 83.59.20.65
C.R.I.N. Bâtiment Loria, B.P. 239
F54506 Vandœuvre-lès-Nancy Cedex

**Abstract**

This paper discusses the idea that a Case-Based Reasoning (CBR) approach offers a good way of making an information retrieval system evolving and adaptive all along its life cycle. In this purpose we propose an approach combining CBR and information retrieval whose aim is to improve the search strategy and to perform contextual adaptation by managing a memory of retrieval sessions. Information retrieval sessions stand for cases in our approach. Management of the memory of sessions exploits success as well as failure of the information retrieval system. Furthermore, such a system can be viewed as a synergy agent between different categories of users (experts in different areas, novices, etc).

## 1 Introduction

This paper deals with Information Retrieval (IR) and more precisely with the design of adaptive Information Retrieval Systems (IRS). Exploiting IRS's experience is of obvious interest to mitigate the preliminary knowledge acquisition bottleneck. A review of the recent literature convinced us that Case-Based Reasoning approach type seems to be an appropriate way to make a knowledge-based system evolving and adapting. This is particularly true when the knowledge in hand is incomplete and noisy.

Our initial goal was to answer the question on how we can make an IRS evolve all along its life cycle. This is called *long-term learning* by contrast with *short-term learning* corresponding to the relevance feedback [Salton 83]. To this purpose we work on a methodology to build an evolving IRS integrating CBR concepts with IR ones. This proposal (CABRI'n for CAse-Based Retrieval of Information - Nancy) has a two-fold objective:

- offering help to IRS design. This can be reached by refining and adapting a generic retrieval strategy according to the users' needs and to the document base. This mechanism can be viewed as *design adaptation* [Hinrichs 91]. In Section 3 we propose such a generic and flexible IR process model;
- building and managing a memory of sessions which will constitute the long-term memory (a retrieval session stands for a case). This memory is designed and used primarily for achieving the first objective. In further steps we intend the memory of sessions for inductive learning purposes.

This will be achieved by taking advantage of the IRS interaction with different categories of users. Users are assumed to be capable of judging the relevance of a proposed document or of the results of a search; more expert users are able to decide whether the retrieval strategy applied to a particular problem is relevant or not. But nobody is good at providing general rules for choosing the right retrieval strategy according to some contextual characteristics of a search situation. Furthermore, the system can be considered as a synergy agent between different categories of users (novices, experts in document retrieval, experts in the document collection domain, ...).

This paper is centered on the idea that CBR is a good way of incrementally improving an IRS. Section 2 discusses motivations for combining IR and CBR. Section 3 is a short description of a parameterized IR process model. The last section describes some characteristic aspects of our approach.

## 2 Combining Case-Based Reasoning and Information Retrieval

It is convenient at this point to discuss the relation between CBR [Riesbeck 89] and IR[1]. Indeed the two fields are related and a mutual contribution is possible. They are similar in the sense that they both try to locate in databases information relevant for a given problem.

Furthermore, IR efforts help CBR in indexing information, in formulating queries to retrieve relevant information, and in defining matching methods.

Conversely, CBR offers a dynamic memory model which allows IR uncertain and incomplete knowledge improvement.

Our approach attempts to exploit analogies as well as mutual reinforcement between CBR and IR.

## 3 Parameterized Information Retrieval Process Model

In IR context documents are poorly indexed i.e, a document index is only a short surrogate of the document itself. Consequently the search strategy is very important to make sure that *precision* (proportion of retrieved items actually relevant) and *recall* (proportion of relevant information actually retrieved) are good. Several learning techniques have been used to improve retrieval system performances. The *relevance feedback* mechanism was primarily proposed to lead to an interactive and iterative retrieval process. The goal is to try to improve *precision* and *recall* values at each step[2] by taking into account the user's relevance assessments (i.e., identification of relevant and non-relevant documents among previously proposed ones) for automatic query reformulation.

Although few formal user experiments have been made on relevance feedback based systems, one can say that even the best ones have a limited recall [Harman 92]. A quite important deal of experimental work has been done to tune different parameters of some existing systems and to evaluate alternative forms.

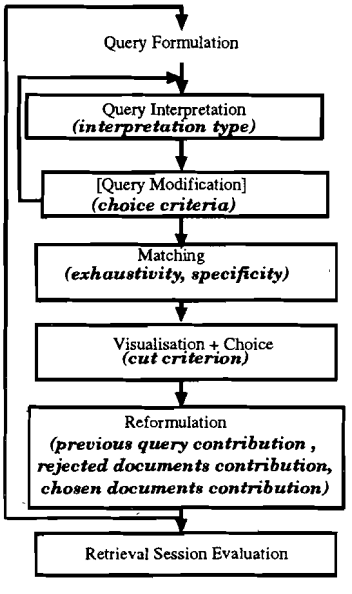



Figure 1: Parameterized information retrieval process model.

Indeed we have often noticed that some choices made during IRS design turn out to be ineffective when using the system. In fact, for each information retrieval primitive, there exist several alternatives (within a given IR model). In Figure 1 we propose a general IR process model. Examples of primitives are: *query interpretation, matching, and reformulation.* Each primitive is parameterized to express the different realization alternatives [Smaïl 93] (parameters are italicized in Figure 1).

[1] This issue was the topic of the AAAI spring symposium held in Palo Alto on march 1993.

[2] A step ranges from query formulation to reformulation.

Furthermore, it has been suggested in the IR literature that different types of user situations, problems, goals, characteristics might require different types of retrieval strategies. This means that besides the generic retrieval process model, we need to have available a typology of potential needs (or queries) that may be addressed. Then for each type of need we have to define a particular instantiation of the retrieval process model. This is performed through what we call *default strategy choice rules.*

# 4 CABRI'n : CAse-Based Retrieval of Information-Nancy

## 4.1 Architecture

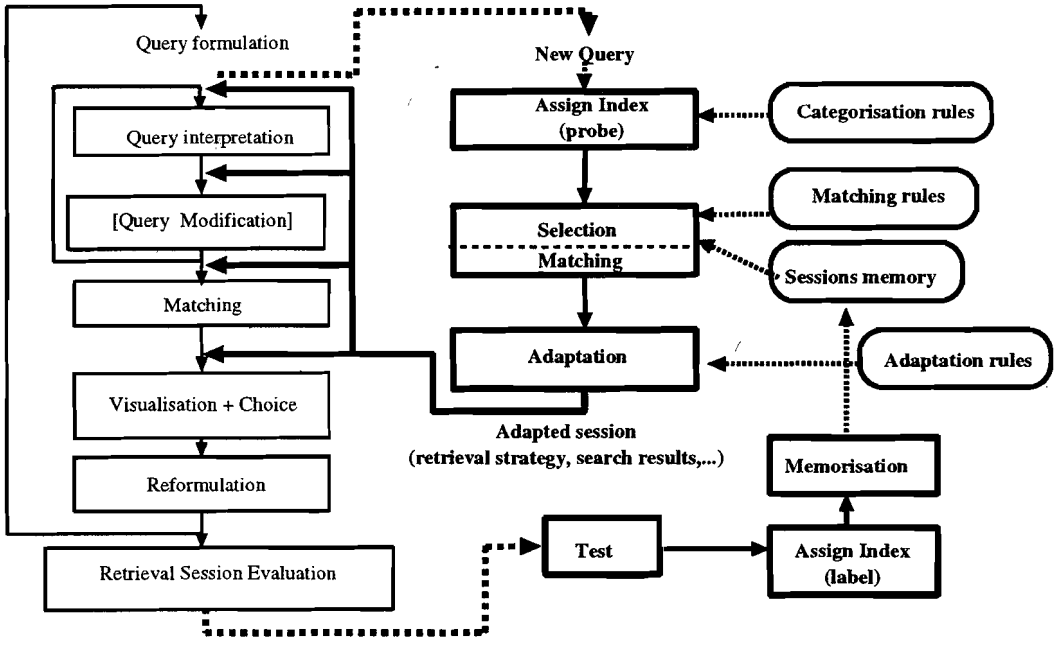



Figure 2: CABRI'n overall architecture.

Figure 2 shows the overall architecture of CABRI'n combining information retrieval (left side) and sessions retrieval (right side). The IRS calls the case-based component like a human intermediary would call for his experience in order to better respond to a query.

To deal with a query, the IRS component uses the particular instantiation of the generic model which is suggested by the CBR part. During the initialization phase, where the memory of sessions is empty or too small, the default strategy choice rules mentioned above (Section 3) will be used to suggest a default instantiation.

When a new information query occurs, it is categorized according to the queries typology invoked in Section 3, before the sessions memory is searched. The query or need type is an important contextual characteristic for former sessions retrieval and adaptation.

For example, needs (expressed in queries) in iconographic databases can be categorized in four types: exploratory need, precise need, connotative need, and thematic need.

The matching step results in a ranked subset of candidate sessions of the same need type as the current one.

Whenever a session similar to the current problem is retrieved it is adapted in a way depending on the need type and relevance level of the retrieved session.

According to the adaptation performed, the IRS takes control at different points (see on Figure 2 arrows originating from Adaptation).
The retrieval session evaluation determines the relevance level of the ending search. The richer this evaluation is, the more finely the current session can be adapted in the future.

### 4.2 Retrieving Search Sessions

A *session* is modeled as an object with two main attributes: *index* and *content*. The index is composed of a *thematic* index and a *global* index. The global index of the session describes its relevance (determined by the session evaluation primitive in Figure 2) and the type of the need it deals with. The session content itself has two levels of detail to allow different types of adaptation. The detailed content includes the session trace whereas the synthesized content involves items related to the adopted strategy, the relevant document set, and the non-relevant document set.

The sessions retrieval is made in two steps. First a selection is performed based on (a part of) the global index; the sessions which correspond to the current need type are thus selected.
Once a selection is done among the memory of sessions, a matching has to be performed between the current query and each selected session. The matching process is based on the thematic index ranking the selected candidates according to the similarity of their search criteria and the current ones.

Furthermore, in order to allow another kind of sessions search based on strategy, we consider strategy as an additional indexing structure (i.e., given a strategy we can find all the sessions which use it). More precisely, a *retrieval-strategy* object aggregates a strategy definition $s$ (parameters of each IR primitive), a collection of sessions using $s$, and a collection of retrieval strategies adapted from $s$.

### 4.3 Adapting Search Sessions

The adaptation function primarily depends on the relevance level of the "best" recalled session.

Adapting a session which led to a failure consists in anticipating this failure in order to avoid it.
If the session to adapt is of mitigate level, the adaptation goal is to propose improvement across certain strategic parameters modifications.

Finally a successful session does not really need adaptation but we can reuse its retrieval strategy or even its results (relevant documents).
Furthermore, the results of the searches performed on a set of the (closest) successful retrieved sessions can be combined to summarize the IRS experience on the current topic. A sampling can be made on these results before presenting them to the user for instance. This retrospective combination is expected to improve the information retrieval recall.

The definite adaptation to perform in each of the three enumerated situations depends secondarily on the current query type.
The described functions have to be refined and we are currently working on the partial reuse of the parameterized IR process (Section 3) for the search sessions retrieval.

## 5 Concluding Remarks and Perspectives

The proposed approach is expected to facilitate the acquisition process of different types of information retrieval knowledge (strategic knowledge, domain knowledge, and organization knowledge).

We are currently implementing a prototype based on the ideas presented above. This is performed in an object-oriented environment (Smalltalk-80) and the document base is an image base. Besides, we are also thinking better of the mutual contribution between IR and CBR in CABRI'n by sharing indexing structures and sharing retrieval primitives.

Up to now, we have assumed that the IRS component could call the CBR part (see Figure 2) only once during a sessions retrieval and that the retrieval strategy was the same during the whole session. An interesting investigation issue would be to make CABRI'n more *reactive* in such a way that IRS could cope with each retrieval step by calling the CBR part. This ambition implies that the CBR retrieval function will have a supplementary search criterion: *similar search evolution*.

In further steps we intend the memory of sessions for two inductive learning purposes. The first consists in extracting explicit knowledge items to enrich the IRS domain knowledge such as contextual thesaurus links and multi-criteria document base organization. The second purpose is to synthesize the adapted strategy over the memory of sessions.

## References


[Harman 92] D. Harman. Relevance Feedback Revisited. In *Proceedings of the Annual International ACM SIGIR Conference*, pages 1–10, Copenhagen, 1992.

[Hinrichs 91] T.R. Hinrichs and J.L. Kolodner. The Roles of Adaptation in Case-Based Design. In *AAAI Conference*, pages 28–33, 1991.

[Riesbeck 89] C.K. Riesbeck and R.C. Schank. *Inside Case-Based Reasoning.* Lawrence Erlbaum Associates Publishers, Hillsdale, New Jersey, 1989.

[Salton 83] G. Salton and M.J. Mac Gill. *Introduction to modern information retrieval.* Mac Graw Hill Book Company, New York, 1983.

[Smaïl 93] M. Smaïl and M. Créhange. Adaptation par cas des stratégies de Recherche d'Information. In *Neuvièmes journées sur les Bases de Données Avancées*, Toulouse, 1993. To appear.

# Retrieving Adaptable Cases: The Role of Adaptation Knowledge in Case Retrieval

Barry Smyth
Hitachi Dublin Laboratory,
Trinity College Dublin,
Dublin, IRELAND.
Phone: +353-1-6798911
EMail: barry@hdl.ie

Mark T. Keane
Computer Science Department,
Trinity College Dublin,
Dublin, IRELAND.
Phone: +353-1-7021534
EMail: mkeane@cs.tcd.ie

**Abstract.** The retrieval of a suitable case is of crucial importance to the success of case-based reasoning. A good criterion for judging "case suitability" is how complex a case will be to adapt. However, it has proven difficult to directly calculate this measure of case "adaptability" without incurring the full cost of adaptation. This has led most researchers to abandon direct *algorithmic* methods in favour of more efficient, albeit less accurate, *heuristic* methods.

This paper describes an approach to case retrieval that allows case adaptability to be accurately measured whilst overcoming the problems which, in the past, led to the adoption of heuristic methods. We argue that this approach benefits from improved retrieval accuracy, flexibility, and greater overall problem solving efficacy. Our methods are implemented in Déjà Vu, a case-based reasoning system for software design, and we use examples from Déjà Vu to demonstrate our approach.

## 1 Introduction

Case-Based Reasoning (CBR) is a reasoning method that exploits experiential knowledge, in the form of past cases, to solve problems [1]. When faced with a new problem, a CBR system will retrieve a case that is similar, and, if necessary, adapt it to provide the desired solution. Obviously, the success of case-based problem solving is crucially dependent on the retrieval of a suitable case; that is, one that can be adapted to give the desired solution. Moreover, the efficiency of case-based methods depends critically on the retrieval of a case that is the easiest, of those available, to adapt.

The majority of CBR systems have proven successful in judging the general suitability of cases to new problem situations. However, accurately determining the "ease of adaptation" or "adaptability" of a given case has proven more difficult because of inherent efficiency problems; how can adaptation be accurately predicted without actually performing the adaptation itself? This has led most researchers to abandon such *deep algorithmic* methods of computing case adaptability in favour of more efficient, albeit less accurate, *shallow heuristic* methods; the hope being that heuristic manipulation of good predictive indices will result in the retrieval of the appropriate case. Typically, these heuristics are designed to give preference to those cases which contain features that have been observed to yield desirable retrieval results. Unfortunately, they seldom anticipate all adaptation problems and less than optimal cases are often retrieved.

In this paper we advance a case selection technique which *can* accurately determine the ease of adaptation of a case while, at the same time, overcoming the efficiency problems that led to the adoption of heuristic methods. The technique uses adaptation knowledge during case selection to "look ahead" to the adaptation stage, allowing its complexity to be assessed, but without incurring the full cost of adaptation. Our methods are implemented in Déjà Vu, a case-based reasoning system for "real world" software design, and we demonstrate our approach using examples from this system. The next section introduces Déjà Vu, detailing the structure of its adaptation knowledge. Section three describes how this knowledge is used in retrieval and includes a very brief review of some conventional heuristic retrieval approaches. Finally, in section four, we argue that our methods benefit from improved retrieval accuracy and flexibility, as well as greater overall problem solving performance.

## 2 Déjà Vu

Déjà Vu is a CBR system for software design operating in the domain of Plant-Control software [2]. Using a hierarchical approach to design, Déjà Vu retrieves a number of cases at different levels of abstraction. These are adapted to provide solutions to the various sub-tasks of the target problem, the resulting solution segments being integrated into the overall solution "on the fly". Problem solving activity is efficiently co-ordinated using a blackboard architecture with dedicated knowledge sources handling the various problem solving stages of analysis, problem decomposition, retrieval, adaptation, and solution integration. Of particular importance, in the context of this paper, is the nature of Déjà Vu's adaptation knowledge which is used during retrieval to improve retrieval accuracy and overall problem solving efficiency.

### 2.1 The Plant Control Domain

Plant-Control software is concerned with controlling autonomous vehicles within a factory or plant environment. Figure 1 illustrates an important class of Plant-Control tasks aimed at the control of vehicles during the loading and unloading of metal coils in a steel mill. Déjà Vu's cases are software modules for controlling vehicles and other devices during such tasks. For example, a simple software design is concerned

with controlling the movement of a coil-car (vehicle) across the factory floor, including collision avoidance, and speed control of the vehicle.

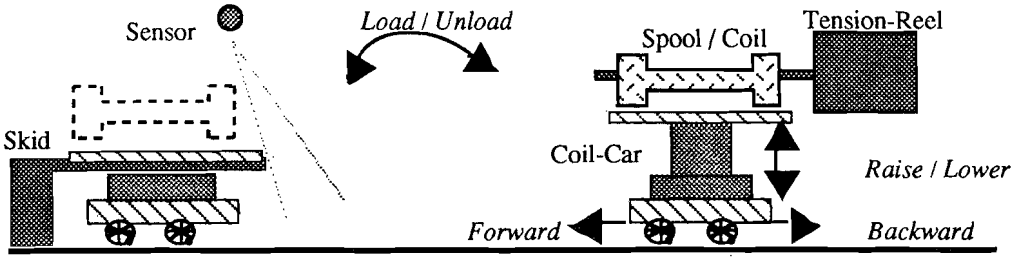



Figure 1. Load/Unload Plant-Control Tasks

## 2.2 Déjà Vu's Adaptation Knowledge

Déjà Vu uses a distributed adaptation scheme that facilitates both specific local modifications, through the action of *adaptation specialists*, as well as global conflict resolution, via *adaptation strategies*. As such adaptation knowledge is captured as a set of specialists and a set of general strategies.

### 2.2.1 Specialists

Adaptation specialists correspond to packages of procedural knowledge each concerned with a specific adaptation task. Each specialist can thus make a specific local modification to a retrieved case. During adaptation many specialists will act on the retrieved case to transform its solution into the desired target solution. Thus, through specialist activity, the differences between the retrieved case and the target are reduced in a fragmentary fashion.

For example, in the plant-control domain, one common difference between a retrieved case and a target problem is that the speed capability of the target's vehicle may differ from that of a retrieved case. To cater for this situation Déjà Vu uses a dedicated *speed specialist* which can satisfy the speed requirements of the target by modifying those of the retrieved case.

As well as procedural knowledge each specialist also has declarative knowledge describing its particular adaptation task. In this way specialists are organised in terms of the modifications they are designed to carry out.

### 2.2.2 Adaptation Strategies

In the course of adapting a retrieved case it is possible that solution conflicts will arise. This is because specialists are not designed to consider the modifications made by others and so interactions that occur between specialists go unchecked. In the past, the resolution of such conflicts has been one of the stumbling blocks of many planning and automated design systems [3]. Déjà Vu attempts to overcome this problem by using an efficient scheme of conflict representation and resolution. Using a set of adaptation strategies, Déjà Vu can detect and repair any conflicts that arise. Strategies are organised in terms of the conflicts they resolve and each is indexed by a description of the type of conflict it can repair. Of course each strategy also has an associated method of repair for resolving the conflict in question.

For example, one common solution conflict occurs when the effect of some event prevents the occurrence of some later event. Figure 2 depicts this situation; the pre-condition state (1) of some goal achieving event (2) has been disabled (or "blocked") by the state (3), a result of some earlier event (4). This *blocked pre-condition* conflict can be resolved by including a new sub-goal which re-enables the blocked pre-condition (1) after the blocking event (4) has occurred. An adaptation strategy to cater for such a conflict would contain a description of this blocked pre-condition situation as well as the appropriate repair method (the inclusion of a sub-goal to re-enable the blocked pre-condition).

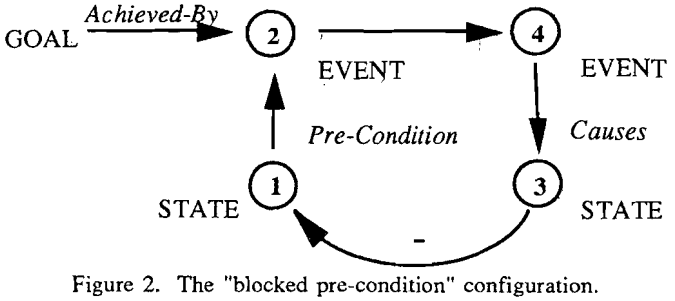



Figure 2. The "blocked pre-condition" configuration.

During adaptation, conflicts can be detected by matching strategy descriptions against the effects of specialists. Specialist activity is described using *influence relations*. These relations are described in more detail in section 3.2.2 but for now let us just say that by using graphs of these influences the qualitative effect of specialist activity can be efficiently characterised, and used as a means of indexing the appropriate adaptation

strategies. A positive match between a strategy description and specialist effects indicates a conflict that can then be resolved by the strategy's repair method.

### 2.2.3 An Example Adaptation Scenario

As an example adaptation scenario consider the following situation. A piece of software is required to move a two speed coil-car to a tension-reel. A case is retrieved which moves a one speed coil-car. Obviously the one speed case must be transformed into a two speed case. Therefore, a speed specialist is used to convert the one speed design into a two speed design. However, such a modification has an adverse affect on the fuel consumption of the coil-car, and the modified design fails because of a lack of fuel. Further modifications are obviously required to remove this conflict between speed and fuel. The detection of such conflicts is where strategy descriptions fit in. The speed increase of a coil-car exerts a negative, blocking influence on its fuel availability. This situation (a blocked pre-condition failure) is captured by the configuration of Figure 2, and the appropriate repair strategy is applied; in this example, the availability of fuel pre-condition is re-enabled by including a re-fuelling stop into the design.

## 3 The Role of Adaptation Knowledge in Case Retrieval

In order to guarantee the retrieval of a case that is the easiest to adapt, the retrieval mechanism must give explicit consideration to *how* cases will be adapted. This is clearly difficult without actually performing the adaptation.

We can think of the processes of retrieval* and adaptation as searching of two distinct search spaces, the specification space and the adaptation space, respectively. To determine the adaptation requirements of a candidate case, a measure of the closeness of the target and candidate in the adaptation space is needed. However, for reasons of efficiency, conventional systems use heuristic rules that select cases on the basis of their closeness in the specification space; that is, cases are compared in terms of specification similarities rather than their more complex (and more important) solution similarities. The hope is that, if two cases have similar specifications then they will have similar solutions, and thus require little adaptation. Unfortunately, this assumption does not always hold, inevitably leading to sub-optimal retrievals.

Déjà Vu's approach to retrieval is different. It uses actual adaptation knowledge during retrieval, to assess specification similarities *directly* in terms of their adaptation requirements, and hence judges a case's suitability by considering how it will be adapted; whereas conventional systems use heuristic rules to select cases, Déjà Vu use rules that are more algorithmic in nature. Essentially the specification space and the adaptation space are coupled by this adaptation knowledge (see Figure 3). Using adaptation knowledge in this way it is possible to predict how specification similarities and dissimilarities will impinge on adaptation by determining how elements of the specification space relate to elements of the adaptation space. Thus, complex adaptation requirements can be determined by comparing the specification's of the target and candidate.

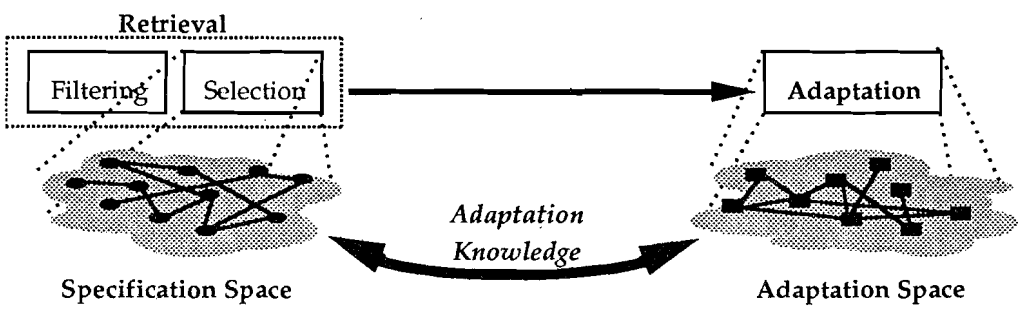



Figure 3. Adaptation knowledge links the specification and adaptation spaces.

### 3.1 Conventional Approaches to Retrieval

Conventional, heuristic approaches to case selection attempt to *estimate* the similarity between the specification of the target and the specification of the candidate case. Exactly *how* a candidate will be adapted is ignored, efficiency being chosen in favour of accuracy. Essentially, the retrieval stage and adaptation stage are de-coupled and the closeness of the target to the candidate case in the adaptation space is estimated by their perceived closeness in the specification space. The rationale being that the case whose specification is most *semantically similar* to the target's will also be the "most useful" case and will require the least adaptation [6, 7].

While such traditional retrieval techniques can produce efficient retrieval results the rationale on which they are based may not be fully justified, and this may ultimately lead to a sub-optimal adaptation stage. That is, the most similar case to the target problem may *not* be the most useful, or indeed, the easiest to adapt. Semantic

* Retrieval can be viewed as a two stage process. First, the filtering stage identifies a small number of candidate cases that are deemed to be contextually relevant to the target. Next, the selection stage performs a detailed analysis between the target and each of these candidates. During this analysis, a set of correspondences or mappings is established between the target and the candidates [4, 5]. In general these mappings are used to determine a measure of similarity between the cases and form the basis of the subsequent adaptation process.

similarity does not guarantee the best results. Two cases could be equally similar to a target problem on this measure and one could be adapted with ease while the other may be considerably harder or even impossible to adapt. To compensate, many researchers have therefore argued that other factors as well as semantic similarity need to be used in retrieval [1, 8, 9, 10, 11], the spirit of this approach being that *all mappings are not equal.*

For example, Kolodner [10] has argued that some mappings found between a target problem and a candidate case should be preferred over others if they exhibit certain characteristics; for instance, if a match is more *specific* or *goal-directed* it should be preferred. In particular, Kolodner also argues that the *ease-of-adaptation* of a match should result in it being preferred over other matches which are indicative of more difficult adaptations. Similarly, Goel's KRITIK system [9] also prefers candidate cases which are easier to adapt by preferring matches which satisfy the functional specifications of the desired, target design. Birnbaum et al. [12] propose a system that learns to index cases on the basis of their adaptability, overriding semantic similarity where appropriate. During problem solving certain features are identified as particularly problematic and cases with such features can be avoided in future problem solving episodes.

In all of these approaches the quality of a candidate case is based on the presence or absence of certain features which are pre-classified as important with respect to retrieval. The relation between specification features and the subsequent adaptation phase is ignored. Consequently, cases are selected on the basis of an "educated guess" rather than through any real insight into their adaptation requirements.

### 3.2 Déjà Vu's Approach to Retrieval: Adaptation Guided Retrieval

In contrast to the above methods, Déjà Vu's retrieval stage gives *explicit* consideration to how a case will be adapted. The retrieval and adaptation stages are coupled by allowing the use of algorithmic adaptation knowledge during retrieval; mappings between the target's and candidate's specifications can be linked directly to elements of the adaptation space. Consequently, specification space closeness can be measured in terms of adaptation space closeness.

Furthermore, the structure of Déjà Vu's adaptation knowledge allows the adaptation requirements of a case to be assessed in an efficient manner, and so the problems that led to the adoption of heuristic approaches in the past are no longer an issue. The result is a more accurate and flexible retrieval stage.

During retrieval, each candidate is judged in terms of the modifications that it would need should it be retrieved. More precisely, case elements that require modification are associated with the adaptation procedures (specialists or strategies) that can perform this modification. In this way it is not only possible to anticipate adaptation success during the retrieval stage, but it is also possible to calculate the complexity of this adaptation.

#### 3.2.1 Specialist Associations

Conventional retrieval systems generate correspondences (mappings) between the target's features and the candidate's features. Normally, these mappings are established according to some measure of *perceived* similarity between the features involved. In contrast, Déjà Vu constructs mappings if and only if there is evidence that the differences that they entail can be correctly adapted.

Déjà Vu's approach is based on the fact that the mappings established between the candidate and target are suggestive of the differences that exist between the candidate solution and desired target solution. Identical mappings suggest candidate solution sections which can be transferred intact to the target. On the other hand, non-identical mappings are indicative of candidate solution sections that will need to be adapted.

In the example of 2.2.3 a non-identical mapping would have been formed between the single speed feature of the candidate and the two speed feature of the target. This mapping served to point out that the candidate solution required a speed modification. To form such a mapping, Déjà Vu requires evidence that the corresponding solution differences *can* be successfully catered for. This evidence exists in the form of specialists. During case selection, sets of mappings are matched against the descriptions of specialists which are designed to perform the entailed modifications. To facilitate the efficient location of the appropriate specialists, the specialist descriptions themselves are in the form of generalised groups of mappings.

#### 3.2.2 Strategy Associations

Like specialists, adaptation strategies are also used during retrieval. As discussed in section 2.2.2, "blind" specialist activity can lead to solution conflicts which must be repaired. Therefore, in predicting the adaptation requirements of a case it is not sufficient to simply determine the appropriate set of specialists without considering the type of conflicts that may arise. To predict conflicts we must be able to describe the effects of specialist activity. This is achieved with the aid of *influence relations* [13].

An influence relation is a qualitative causal relationship between two domain elements. It specifies that one element (the *influencer*) effects another (the *influenced*) in some way. The mode of influence can be either positive (+) or negative (-). A positive influence means that a change in the influencer entails a corresponding change in the influenced. For example, speed and fuel consumption are connected by a positive influence relation from speed to fuel consumption; an increase in speed leads to an increase in fuel consumption. A negative influence means that a change in the influencer leads to a qualitatively opposite change in the influenced. For example, fuel consumption exerts a negative influence on fuel availability; an increase in fuel consumption causes less fuel to be available. Using graphs of these influence relations an qualitative model of

the dependencies between domain elements can be built up. With these graphs it is possible to describe both the desired effects and side-effects of specialists. For example, the speed specialist changes the speed of a case. According to the influences above it also effects the fuel consumption and fuel availability of the case.

Strategies are indexed into the domain knowledge-base by sets of influence relations. During retrieval the specialist associations activate a set of influences that capture their intended effect. In turn these influences activate relevant strategy descriptions, indicating possible conflict problems. The retrieval context is used to instantiate these strategies which are then associated with the problematic specialists and mappings. In this way, during retrieval, solution conflicts can be predicted and repairs scheduled.

### 3.2.3 An Example Retrieval Scenario

As an example, let us return to the problem of section 2.2.3 which was to design a two speed movement case from a single speed case. We saw the type of modifications that are necessary in this adaptation scenario. Now we demonstrate how these modifications are predicted during retrieval.

The mappings between the speed features of the candidate and target signify the need for a speed modifying specialist. Once a specialist has been found the mapping can be established. In addition, a measure of the quality of the mapping is based on the computational complexity of the specialist. But, what about predicting conflicts? In particular, how can the fuel availability problem be foreseen and an appropriate strategy identified to effect its repair ?

The target problem is concerned with moving a two speed coil-car to a tension-reel (1). A pre-condition of movement is that fuel be available (2). The speed specialist will case the speed of the case to be increased. The influence that this increase in speed (3) exerts on the consumption of fuel (4) leads to the disablement of the fuel availability pre-condition. This configuration (boxed portion of Figure 4) matches the description for the blocked pre-condition strategy of section 2.2.2. After instantiating the strategy in the current context (unboxed region of Figure 4) it is associated with the speed specialist. During adaptation the action of the speed specialist is augmented with the repair action of this adaptation strategy; in this case adaptation consists of changing the speed of the case and adding a re-fuelling stop.

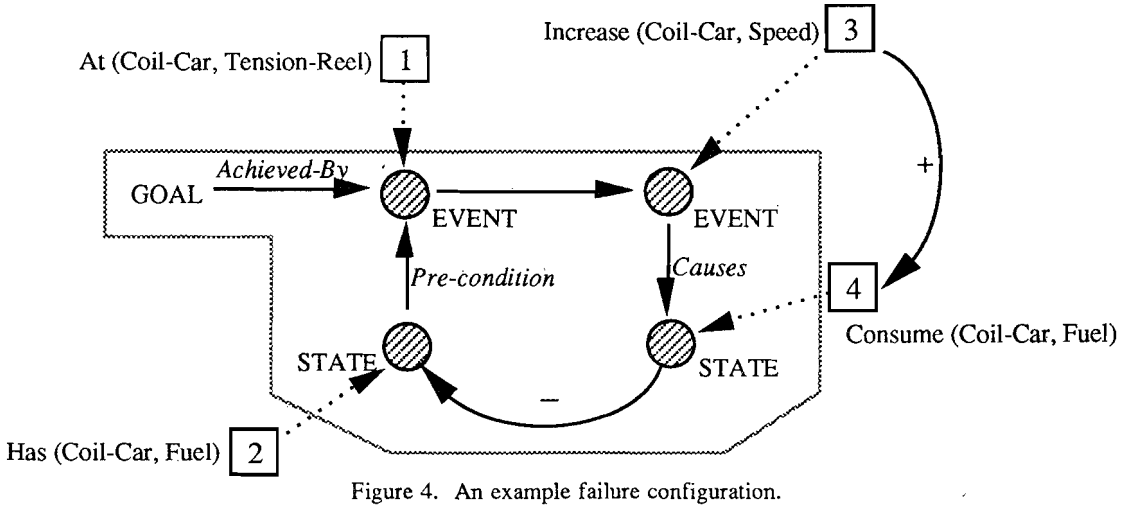



Figure 4. An example failure configuration.

Although simple, the example above does highlight the key features of our approach: the relevant local and global adaptation knowledge (in the form of specialists and adaptation strategies) is efficiently assembled during retrieval enabling an accurate judgement to be made on the adaptation requirements of a candidate case.

## 4 Beneficial Implications

Our approach ensures the retrieval of a case that requires minimal adaptation. This is in contrast to other CBR systems that do not directly couple retrieval and adaptation and, as such, can only estimate the usefulness of a given case in terms of its semantic similarity -- which is often not a very accurate measure of adaptability.

Retrieval now carries out the preliminary adaptation work by identifying and instantaiting the specialists and strategies that will be necessary during the adaptation stage. The additional retrieval complexity which this involves is minimised by organising adaptation knowledge in a manner that permits the efficient identification of the appropriate specialists and strategies. Moreover, any additional retrieval expense is offset by improved adaptation efficiency; not only is some of the adaptation work carried out during retrieval, but the case retrieved should be the easiest, of those available, to adapt.

In addition, greater retrieval flexibility is also achieved. With conventional approaches, changes to the adaptation capabilities of a system will not be immediately reflected in the retrieval preferences of the system. Instead changes must be made to the retrieval heuristics to capture the new adaptation possibilities. In contrast, because the retrieval and adaptation stages are directly coupled in Déjà Vu, any changes to its adaptation

capabilities *will* be immediately available to the retrieval system; the altered adaptation knowledge itself is used in retrieval.

Finally, the representational requirements of the approach are domain independent and thus facilitate the adoption of the technique across a range of CBR application domains.

## 5 Conclusions

The main thrust of the paper centres on the description of an important issue in CBR, that of case selection. More precisely, it concentrates on a critical case selection criterion, that of adaptation efficiency. Through Déjà Vu's coupling of the retrieval and adaptation processes, an efficient model of this selection criterion is realised. Succeeding where similar methods have failed in the past, an approach is described that can perform accurate and efficient algorithmic assessments of the adaptation requirements of retrieval candidates, without incurring the full cost of adaptation. Researchers have abandoned such algorithmic selection methods in the past because of the difficulty in predicting the potentially complex set of interactions that can arise during adaptation. Déjà Vu has tackled these interaction issues head-on by advancing a mechanism for resolving such conflicts during adaptation while facilitating their prediction during retrieval.

The result is an approach to retrieval which, improves retrieval accuracy and flexibility as well as overall problem solving performance, and can be applied to a range of case-based reasoning tasks.

## 6 References


1. K.J. Hammond: Planning from Memory. In: B. Chandrasekaran (eds.): Case-Based Planning. Academic Press 1989, pp. 66-70

2. B. Smyth, and P. Cunningham: Déjà Vu: A Hierarchical Case-Based Reasoning System for Software Design. In: B. Neumann (eds.): Proceedings of the 10th European Conference on Artificial Intelligence. Wiley 1992, pp. 587-589

3. J. Hendler, A. Tate, M. Drummond: AI Planning: Systems and Techniques. AI Magazine, **11**(2), 61-77 (1990)

4. M.T. Keane: Analogical problem solving. Chichester: Ellis Horwood, 1988 (Halstead Press, Wiley in N.America)

5. D. Gentner: Structure-mapping: A theoretical framework for analogy. Cognitive Science, **7**, 155-170 (1983)

6. R. Bareiss, J.A. King: Similarity Assessment in Case-based Reasoning. In: K.J. Hammond (eds.): Proceedings of the Case-Based Reasoning Workshop. Morgan Kaufmann 1989, pp. 67-71.

7. P. Thagard, K.J. Holyoak, G. Nelson, D. Gochfeld: Analog Retrieval by Constraint Satisfaction. Artificial Intelligence, **46**, 259-310 (1990)

8. T. Cain, M.J. Pazzani, G. Silverstein: Using Domain Knowledge to Influence Similarity Judgements. In: R. Bareiss (eds.): Proceedings of the Case-Based Reasoning Workshop. Morgan Kaufmann 1991, pp. 191-198

9. A.K. Goel: Integration of Case-Based Reasoning and Model-Based Reasoning for Adaptive Design Problem Solving. Ph.D. Dissertation. Ohio State University (1989)

10. J. Kolodner: Judging Which is the "Best" Case for a Case-Based Reasoner. In: K.J. Hammond (eds.): Proceedings of the Case-Based Reasoning Workshop. Morgan Kaufmann 1989, pp. 77 - 84

11. E.L. Rissland, K.D. Ashley: Credit Assignment and the Problem of Competing factors in Case-Based Reasoning. In: J. Kolodner (eds.): Proceedings of the Case-Based Reasoning Workshop. Morgan Kaufmann 1988, pp. 327 - 344

12. L. Birnbaum, G. Collins, M. Brand, M. Freed, B. Krulwich, L. Pryor: A Model-Based Approach to the Construction of Adaptive Case-Based Planning Systems. In: R. Bareiss (eds.): Proceedings of the Case-Based Reasoning Workshop. Morgan Kaufmann (1988), pp. 191 - 198

13. K.P. Sycara, D. Navinchandra: Influences: A Thematic Abstraction for Creative Use of Multiple Cases. In: R. Bareiss (eds.):Proceedings of the Case-Based Reasoning Workshop. Morgan Kaufmann (1991), pp. 133 - 144

# Improving the Retrieval Step in Case-Based Reasoning*

Stefan Wess, Klaus-Dieter Althoff and Guido Derwand
University of Kaiserslautern
D-67653 Kaiserslautern, Germany
wess@informatik.uni-kl.de

**Abstract**

Retrieval of cases is one important step within the case-based reasoning paradigm. We propose an improvement of this stage in the process model for finding most similar cases with an average effort of $O[log_2 n]$, $n$ number of cases. The basic idea of the algorithm is to use the heterogeneity of the search space for a density-based structuring and to utilize this precomputed structure for efficient case retrieval according to a given similarity measure. Therefore, we combine basic aspects of object-oriented data bases, information retrieval, nearest neighbor classification, and case-based reasoning for the improvement of well-known techniques. The described approach is fully implemented and currently used in a case-based reasoning system (PATDEX) for diagnostic applications in technical domains.

## 1 Introduction

Retrieval of sufficiently similar cases is one of the main points in the process model of case-based reasoning, i.e. before selecting the most useful case(s) for adaptation, the case base must be restricted to a small set of reasonable candidates. Retrieval and selection of cases are often distinguished by the kind of features they use for case comparison (*surface* versus *structural* similarity: [16]). To detect really useful cases for the problem at hand, the selection step has to consider all available knowledge of the underlying domain. Thus, computing this structural similarity match is very expensive. Unfortunately, the retrieval step which deals with all cases in the case base must be computed very fast. Therefore, this step can only rely on the comparison of syntactical features (*surface similarity*) [14]. Basically, there are two different approaches to similarity assessment in case-based reasoning [22, 3]: the *representational approach*, proposed by [17] using a structured memory of cases, and the *computational approach* e.g. [25, 1], which is based on the computing of an explicit similarity function (cf. [29]).

A naive approach to case retrieval would be to compute the surface similarity by comparing syntactical features of every case in the case base to the current problem according to a given similarity measure. The set of cases which must be examined by the following selection procedure is then determined by the $m$-most similar cases ($m$ fixed), or by all cases exceeding a given similarity threshold $\delta$. Many known case-based reasoning systems use this simple kind of approach (at least hidden in the implementation). Since the overall complexity of this retrieval procedure is $O[n]$, $n$ number of cases, for small case bases this strategy is reasonable. But, for increasing case bases this procedure leads to a too time-consuming process that restricts this approach to toy domains.

Up to now, the improvement of the efficiency of the retrieval step has been the goal in different research projects. We can distinguish two main approaches: First, the brute-force methods using massively parallel architectures like [25, 19] which take up to one processing element for each case in the case base. Second, precomputation of indices (*cf.* [28]) for rapid access to the case base, e.g. [5, 26, 4]. The first approach needs a lot of hardware support for the speed up of the retrieval process. By using the second approach, it is difficult to guarantee the completeness of the retrieval according to the used similarity measure.

The problem of determining the most similar cases (best matches) based on a given case description is well known as *nearest neighbor search* [8]. Cases can then be interpreted as points within a multidimensional search space where each attribute implements one dimension that can be searched with an associative procedure. The main idea of the proposed approach is to structure the search space based on its observed density and using this precomputed structure for efficient case retrieval according to the given similarity measure. We developed a retrieval mechanism [21] based on a $k$-d tree, a multi-dimensional binary search tree [6, 12, 7]. Within the $k$-d tree an incremental best-match search is used to find the $m$ most similar

*Funding for this research has been partially provided by the Commission of the European Communities (ESPRIT contract P6322, the INRECA project). The partners of INRECA are AcknoSoft (prime contractor, France), tecInno (Germany), Irish Multimedia Systems (Ireland), and the University of Kaiserslautern (Germany).

cases (nearest neighbors) within a set of $n$ cases with $k$ specified indexing attributes (dimensions). The search is guided by application-dependent similarity measures based on user-defined value ranges. The overall similarity measure is split into local measures for each value range and a global measure which is composed from the local ones [23]. A $k$-d tree as such is comparable to a discrimination net [10, 9] that has been optimized for similarity-based retrieval of cases.

Beyond the pure data structure the $k$-d tree approach includes procedures for optimizing the tree structure both from scratch, or incrementally. In addition, search procedures are available that take advantage of the known geometric boundaries along the various indexing dimensions. This is important for a correct search procedure to be efficient. Cases with missing attribute values can also be found in a reasonable amount of time. With respect to the special use of similarity measures in our approach, we are restricted to have a monotic and symmetric global similarity measure, monotonic and symmetric local similarity measures, and ordered value ranges. Nevertheless, similarity measures as described in [4] and learning of improved similarity measures as described in [24] can be applied. Therefore, the proposed approach can be seen as a natural and reasonable extension of the PATDEX system, a case-based reasoning system for diagnostic applications [30]. PATDEX is an integrated subpart of the knowledge acquisition workbench MOLTKE [2] including systems for heuristic diagnosis as well as inductive and model-based reasoning. PATDEX uses knowledge-based methods to improve its similarity estimations. Therefore, it is able to process (among others) default values for symptoms, heuristic determination rules for symptom values being generated by an inductive learning system, and causal determination rules being generated by a knowledge compilation system. PATDEX can also handle abnormal and unknown symptom values. The main restriction of PATDEX is that its indexing mechanisms can only deal with symbolic value ranges and the processing of very large case bases could be a problem if there exist too many attribute values. Therefore, a multi-dimensional retrieval structure, namely a $k$-d tree, is used to overcome these problems.

The associative search mechanism, as proposed below, is used for the basic indexing and retrieval task [12, 7], but has to be seen in the broader context of a real complex application [4]. Therefore, many additional improvements have been implemented [21]. The improvements encompass the following: different weightings for the respective attributes, several different similarity measures within the same tree, learning of improved similarity measures, different kinds of predefined symbolic (local) similarity measures as well as handling of incomplete or missing data. Additionally, the matching of object-oriented case representations instead of flat feature-based attribute vectors is included. For reasons of efficiency, the retrieval procedure is built on top of an object-oriented data base (GEMSTONE).

## 2 Building a $k$-d Tree

The basic idea of the approach is to build a tree which splits the search space into parts which contain a number of similar cases according to the given similarity measure (Figure 1).

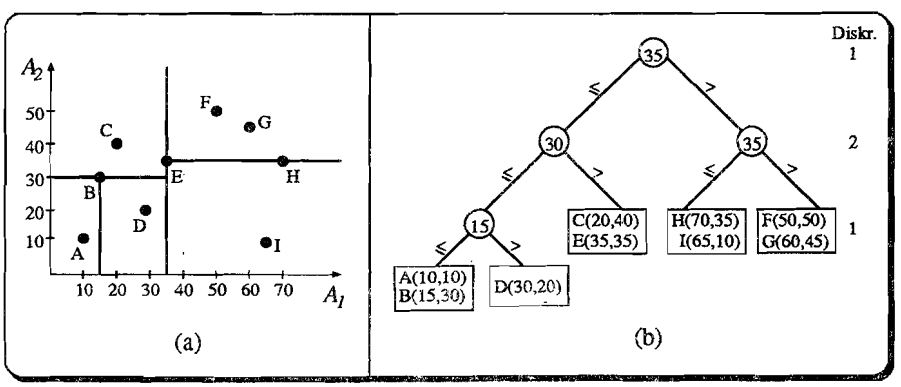



Figure 1: An exemplary two-dimensional search space and the according $k$-d tree

Therefore, every node within the $k$-d tree represents a subset of the cases of the case base and the root node represents the whole case base. Every inner node partitions the case set into two disjoint subsets, storing the bounding values for each dimension (attribute). The leaves of the tree which contain a specific number of cases are called buckets. For the construction of the tree, we have to choose the best partitioning attribute which divides the case base into two equally sized parts [12]. The process continues recursively for each of the constructed subsets of the case base until only a few cases (bucket size) remain which are stored together in one bucket. The determination of the partitioning attribute (dimension) is the most crucial part of the approach. For best speedup of the retrieval process the partition of the search space has to reflect the structure and the density of the underlying case base.

To estimate the dispersion, we use a statistical measure, namely the *interquartile distance* that can be used for both numeric and ordered nominal attribute value ranges.

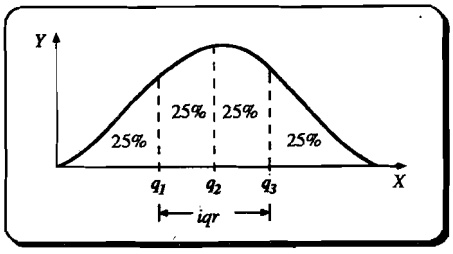



Figure 2: Basic idea of the partitioning

While the median splits a given distribution of values into two equally-sized areas, quartiles split them into four (Figure 2). The first quartile $q_1$ (25% quartile) divides the "lower half" of the distribution into two equally-sized areas as the third quartile $q_3$ (75% quartile) does with the "upper half" of the distribution. The median is denoted as the second quartile. The interquartile distance *iqr* is then computed as the distance between the first and the third quartile. The bigger the distance between these quartiles, the bigger is the dispersion of the attribute values. During tree construction that attribute having the maximal dispersion is selected as the discriminating attribute. Since we use similarities and not distances, we want to introduce the *interquartile similarity* as a new term. It denotes that we select that attribute for discriminating purposes where the respective quartiles have the lowest similarity (which corresponds to the maximal distance).

```
procedure BUILD_TREE(setOfData);
local j, disc, minSimilarity, p;
begin
  if Size(setOfData) ≤ b then return MAKE_TERMINAL_NODE(setOfData);
  minSimilarity:=∞;
  for all coordinates A_j (1 ≤ j ≤ k) do
    if SPREAD(A_j,setOfData) < minSimilarity then
    begin
      minSimilarity:=SPREAD(A_j,setOfData);
      disc:=j
    end;
  p:=MEDIAN(disc,setOfData);
  return
    MAKE_NONTERMINAL_NODE
      (disc,p,
      BUILD_TREE(LEFT_SUBFILE(disc,p,setOfData)),
      BUILD_TREE(RIGHT_SUBFILE(disc,p,setOfData))
      );
end BUILD_TREE.
```

The procedure SPREAD($A_j$,*setOfData*) computes the dispersion of the values of attribute $A_j$ for the set of data *setOfData* using the interquartile similarity. The procedure MEDIAN(*disc*,*setOfData*) computes the median of the discriminating attribute *disc* based on the values of *disc* given by *setOfData*. LEFT_SUBFILE and RIGHT_SUBFILE generate the two partitions of *setOfData* with respect to the discriminating attribute *disc* and the discriminating value $p$. MAKE_TERMINAL_NODE and MAKE_NONTERMINAL_NODE generate leaf nodes and inner nodes, respectively. Every leaf node contains within its *bucket* at most b cases where b is the predefined bucket size. An inner node contains its discriminating attribute *disc*, the respective discriminating value $p$ as well as two pointers to its left and right successor node (*leftSon* and *rightSon*).

The average case effort [20] for generating a $k$-d tree is $O[k * n * log_2 n]$, for the worst case $O[k * n^2]$. The average costs for retrieving the most similar case are $O[log_2 n]$, if the tree is optimal organized. For the worst case, the retrieval costs are $O[n]$. The retrieval mechanism is correct and complete in the sense that it always returns the $m$ most similar cases according to the specified global similarity measure *sim*.

# 3 Searching Similar Cases using a $k$-d Tree

The search for similar cases in the $k$-d tree is done via a recursive tree search procedure according to the global similarity measure *sim*. Normally, there are no fully identical cases in the case base and we have

to look for the most similar ones. Using the tree as a kind of binary search tree leads to a bucket where a specific number of cases are stored. At this stage, it is necessary to compute the similarity of each case stored in the bucket using the predefined similarity measure *sim*. If we are looking for the $m$ most similar cases we can build up a queue containing these most similar cases. Using this queue we draw a hyperball around the given problem that includes the $m$ most similar cases found in the current bucket. Thus, every case which is at least as similar as the examined ones must be within this constructed $k$-dimensional hyperball.

Figure 3 describes the basic idea. In this example, we have a current problem called $X_q$ (query) and up to four similar cases $PQC[1] \ldots PQC[4]$ found in one bucket. Cases at least as similar as $PQC[4]$ like $X$ but not examined yet appear also in the 2-dimensional ball. The single point outside of the ball is not similar enough and has not to be considered.

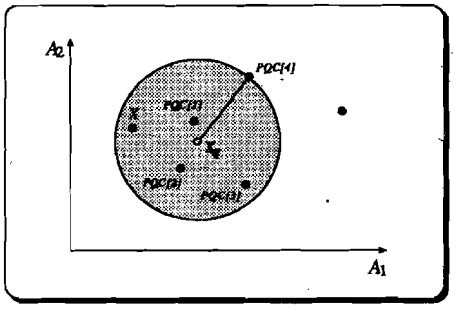



Figure 3: Basic idea of the bounds tests: building a hyperball

For an efficient implementation of this basic idea we use two test procedures [12]: BALL-WITHIN-BOUNDS (BWB) and BOUNDS-OVERLAP-BALL (BOB) (Figure 4). These procedures check whether it would be reasonable to explore certain areas of the search space in more detail, or not. Such tests can be carried out without retrieving the respective cases. The geometric bounds of the considered subspaces are used to compute a "similarity interval" whose upper bound then "answers" the question to explore, or not. For finding the m most similar cases for a given working case (or query case), we apply recursive tree search. Thus, as input we need the query case $X_q$, the number $m$ of most similar cases, the $k$-d tree represented by its root node, and the global similarity measure *sim*. During search a priority queue $PQC$ is continuously updated which includes the $m$ most similar cases (while PQC[n] denotes the nth most similar case, PQS[n] denotes the actual similarity value of the nth most similar case). If the recursive search procedure examines a leaf node, the similarity of all included cases is computed and, if necessary, the priority queue $PQC$ is updated. If the examined node is an inner node, then the search procedure is recursively called for that son node which should include the query case. If this call terminates, it is tested whether it is also necessary to examine the other son node by using the BOB test.

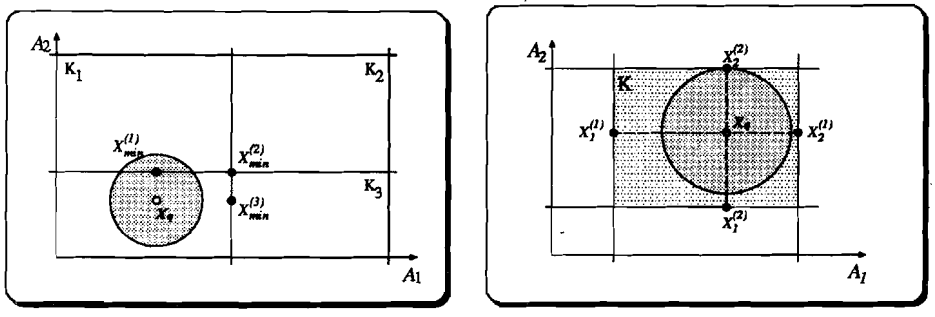



Figure 4: Basic idea of the BOB and BWB test

The BOB test is TRUE if the cases of the actual tree node have to be explored. The inner nodes are correct generalizations of all the cases they represent in the sense that they include the geometric (upper and lower) bounds (for every indexing attribute) which correspond to the respective subspace.

$$BOB \iff Sim(X_{min}, X_q) \geq PQS[m] \; ( = Sim(PQC[m], X_q) \; )$$

These geometric bounds are used to compute a similarity interval whose upper bound then answers the question to explore, or not. The closest point $X_{min}$ within the actual node's subspace is computed as the

projection onto the actual node's geometric bounds (Figure 4). $X_{min}$ is on the actual node's bounding box on the edge facing the query case $X_q$. If there is no overlapping in any of the $k$ dimensions between the node's bounding box and the $k$-dimensional ball round $X_q$, then $X_{min}$ is a corner of the bounding box. If $X_q$ is within the bounding box then $X_q = X_{min}$ (Figure 4). Before the recursive search procedure terminates, the BWB test is applied. This test is TRUE if the $k$-dimensional ball round $X_q$ is completely within the bounding box of the actual tree node (Figure 4).

$$BWB \iff Sim(X_1^{(j)}, X_q) < PQS[m] \quad \wedge \quad Sim(X_2^{(j)}, X_q) < PQS[m] \; \forall j = 1, \ldots, k$$

In this case, no overlapping with other bounding boxes is possible. Thus, the search is finished, and the $m$ most similar cases for the current problem according the given global similarity measure *sim* are found.

```
procedure SEARCH(node);
local p, d, temp;
begin
  if isTerminal(node) then (* node is a leaf node *)
  begin
    "test the cases in node.bucket and update PQC, PQS";
    (* test if the search can be finished *)
    if BALL-WITHIN-BOUNDS then done else return
  end;
  d:=node.discriminator;
  p:=node.partitionValue;
  (* recursive call of the son node that contains X_q *)
  if X_q[d] ≤ p then
  begin
    temp:=Upper[d];   Upper[d]:=p;
    SEARCH(node.leftSon);
    Upper[d]:=temp;
  end
  else begin
    temp:=Lower[d];   Lower[d]:=p;
    SEARCH(node.rightSon);
    Lower[d]:=temp;
  end
  (* recursive call of the other son node *)
  if X_q[d] ≤ p then
  begin
    temp:=Lower[d];   Lower[d]:=p;
    if BOUNDS-OVERLAP-BALL then SEARCH(node.rightSon);
    Lower[d]:=temp;
  end
  else begin
    temp:=Upper[d];   Upper[d]:=p;
    if BOUNDS-OVERLAP-BALL then SEARCH(node.leftSon);
    Upper[d]:=temp;
  end;
  (* test if search can be finished *)
  if BALL-WITHIN-BOUNDS then done else return;
end SEARCH.
```

# 4 Summary

Case-based reasoning using a simple case representation and avoiding case adaptation (which is, e.g. the case for case-based fault diagnosis in engineering systems) is comparable to conceptual clustering in the sense that incremental concept formation as described in [13] can be viewed as a special variant of case-based reasoning, i.e. from a very abstract point of view both approaches are identical. As a consequence, these two approaches might benefit from one another if the above assumptions are fulfilled. In fact, we currently experiment with an alternative for determining the partitioning attribute and the dispersion of the attribute values which is based on the COBWEB approach [11]. But, our approach proposed here significantly differs from conceptual clustering because class-dependent information is used to dynamically decide on the selection of an appropriate global similarity measure using the well informed

PATDEX similarity measures. In addition, case retrieval via $k$-d trees is only one subcomponent of the PATDEX system being combined with other techniques. The combination of information retrieval/nearest neighbor classification and CBR has been of increasing interest recently e.g. [27]. Nevertheless, we are not aware of any similar approach which is also correct, complete and efficient.

# References


[1] David W. Aha. Case-Based Learning Algorithms. In Ray Bareiss, editor, *Proceedings: Case-Based Reasoning Workshop*, San Mateo, California, 1991. DARPA, Morgan Kaufmann Publishers. Washington, D.C., USA, May 8–10, 1991.

[2] K.-D. Althoff, F. Maurer, S. Wess, and R. Traphöner. MOLTKE - an integrated workbench for fault diagnosis in engineering systems. In S. Hashemi, J.P. Marciano, and G. Gouarderes, editors, *Proc. 4th international conference Artificial Intelligence & Expert Systems Applications (EXPERSYS-92)*, Paris, October 1992. i.i.t.t international.

[3] K.-D. Althoff and S. Wess. Case-Based Reasoning and Expert System Development,. In F. Schmalhofer, G. Strube, and T. Wetter, editors, *Contemporary Knowledge Engineering and Cognition*. Springer Verlag, 1992.

[4] Klaus-Dieter Althoff and Stefan Wess. Case-Based Knowledge Acquisition, Learning and Problem Solving for Diagnostic Real World Tasks. In *Proceedings EKAW-91*, 1991.

[5] R. Barletta and W. Mark. Explanation-Based Indexing of Cases. In Kolodner [18], pages 50–61. Clearwater Beach, Florida, USA, May 10–13, 1988.

[6] J.L. Bentley. Multidimensional binary search trees used for associative. *Commun ACM*, 18:509–517, 1975.

[7] A.J. Broder. Strategies for efficient incremental nearest neighbor search. *Pattern Recognition*, 23:171–178, 1990.

[8] Belur Dasarathy. *Nearest Neighbor Norms: NN Pattern Classification Techniques*. IEEE Computer Society Press, 1990.

[9] D. McDermott E. Charniak. *Introduction to AI*. Addison-Wesley, 1985.

[10] E.A. Feigenbaum. The simulation of verbal learning behavior. In E.A. Feigenbaum and J. Feldmann, editors, *Computer and Thought*. McGraw Hill, New York, 1963.

[11] D. Fisher. Cobweb: Knowledge acquisition via conceptual clustering. *Machine Learning*, 2:139–172, 1987.

[12] J.H. Friedman, J.L. Bentley, and R.A. Finkel. An algorithm for finding best matches in logarithmic expected time. *ACM Trans. math. Software*, 3:209–226, 1977.

[13] J. H. Gennari, P. Langley, and D. Fisher. Models of incremental concept formation. *Artificial Intelligence*, 40:11–61, 1989.

[14] Dedre Gentner and Kenneth D. Forbus. MAC/FAC: A model of similarity-based retrieval. In *Proceedings of the 13th Annual Conference of the Cognitive Science Society*, pages 504–509, 1991.

[15] Kristian J. Hammond, editor. *Proceedings: Case-Based Reasoning Workshop*, San Mateo, California, 1989. DARPA, Morgan Kaufmann Publishers. Pensacola Beach, Florida, USA, May 31–June 2, 1989.

[16] K. J. Holyoak and K. Koh. Analogical Problem Solving: Effects of Surface and Structural Similarity in Analogical Transfer. In Midwestern Psychological Association, editor, *Proceedings of the Conference of the Midwestern Psychological Association*, 1986. May 1986.

[17] Janet L. Kolodner. *Retrieval and Organizational Strategies in Conceptual Memory*. Lawrence Erlbaum, Hillsdale, New Jersey, 1984.

[18] Janet L. Kolodner, editor. *Proceedings Case-Based Reasoning Workshop*, San Mateo, California, 1988. DARPA, Morgan Kaufmann Publishers. Clearwater Beach, Florida, USA, May 10–13, 1988.

[19] Janet L. Kolodner. Retrieving Events from a Case Memory: A Parallel Implementation. In *Proc. Case-Based Reasoning Workshop* [18], pages 233–249. Clearwater Beach, Florida, USA, May 10–13, 1988.

[20] Kurt Mehlhorn. *Data Structures and Algorithms 3: Multi-dimensional Searching and Computational Geometry*. Springer, 1984.

[21] Hannes Öchsner and Stefan Wess. Ähnlichkeitsbasiertes Retrieval von Fallbeispielen durch assoziative Suche in einem mehrdimensionalen Datenraum. In K-D. Althoff, S. Wess, B. Bartsch-Spörl, and D. Janetzko, editors, *Ähnlichkeit von Fällen in Systemen des fallbasierten Schliessens*, SEKI-Report, Universität Kaiserslautern, SFB 314, 25.-26. Juni, 1992.

[22] B. W. Porter. Similarity Assessment: Computation vs. Representation. In Hammond [15], pages 82–84. Pensacola Beach, Florida, USA, May 31–June 2, 1989.

[23] M. M. Richter and S. Wess. Similarity, Uncertainty and Case Based Reasoning in PATDEX. In Robert S. Boyer, editor, *Automated Reasoning - Essays in Honor of Woody Bledsoe*, pages 249–265. Kluwer Academic Publishers, 1991.

[24] Michael M. Richter. Classification and learning of similarity measures. In *Proc. der 16. Jahrestagung der Gesellschaft für Klassifikation e.V.* Springer Verlag, 1992.

[25] Craig Stanfill and David Waltz. Toward Memory-Based Reasoning. *Communications of the ACM*, 29(12):1213–1229, 1986.

[26] R. H. Stottler, A. L. Henke, and J. A. King. Rapid Retrieval Algorithms for Case-Based Reasoning. In *Proceedings of the 11th International Conference on Artificial Intelligence IJCAI-89*, pages 233–237. IJCAI, 1989. Detroit, Michigan, USA.

[27] Toshikazu Tanaka and Naomichi Sueda. Combining strict matching and similarity assessment for retrieving appropriate cases efficiently. In *Working Notes AAAI Spring Symposium: Case-Based Reasoning and Information Retrieval*, pages 112–119, 1993.

[28] P. Thagard and K. I. Holyoak. Why Indexing is the Wrong Way to Think About Analog Retrieval. In Hammond [15], pages 36–40. Pensacola Beach, Florida, USA, May 31–June 2, 1989.

[29] A. Tversky. Features of Similarity. *Psychological Review*, 84:327–352, 1977.

[30] Stefan Wess. PATDEX - Inkrementelle und wissensbasierte Verbesserung von Ähnlichkeitsurteilen in der fallbasierten Diagnostik. In *Tagungsband 2. deutsche Expertensystemtagung XPS-93*, Hamburg, 1993. Springer Verlag.

# Using a High-Level, Conceptual Knowledge Representation Language for Visualizing Efficiently the Internal Structure of Complex "Cases"

Gian Piero Zarri

Centre National de la Recherche Scientifique
CERTAL - INALCO
73, rue Broca
75013 PARIS, France

**Abstract.** In this short paper, we assert that the implementation of more sophisticated, "second generation" CBR applications can benefit from the adoption of high-level knowledge representation languages like NKRL ("Narrative Knowledge Representation Language"), which can be used to describe the internal structure of complex cases. After having evoked very succinctly the main characteristics of NKRL, we mention briefly the possible use of this language for typical CBR applications like analogical reasoning or indexing.

## 1 Introduction

In the practice of case-based reasoning (CBR) techniques, the "flat structure" paradigm seems to be largely diffused : accordingly, cases are represented simply as flat lists of features without internal structure. On the other hand, even if we are aware of several theoretical proposals — see, e.g., [1] — and prototypical systems — see, e.g., [2] — concerning the use of more structured representations like event concept coherence networks (EEC networks) or frames, there is no agreement on how to impose more complex internal structuring on cases, and for what reasons. This attitude is in good agreement with a well-diffused indifference, in the CBR milieu, towards the knowledge representation problems.

In this short paper, we affirm on the contrary that, (at least) for domains like the socio-economic-political one (according to the broadest meaning of these words), the possibility of putting to work more sophisticated, "second generation" CBR applications can be facilitated by the possibility of making use of more powerful knowledge representation tools. If, on one hand, in a socio-economic-political context, it is difficult to avoid the comparison of information given in the form of cases, examples or stories, it is also evident that the "knowledge" to be represented is, here, too complex and informal to fit well into the "flat" representation which can be sufficient, e.g., for CBR applications in the diagnostic style.

## 2 A Short Description of NKRL

In this Section, we will evoke briefly the main characteristics of a high-level, implemented, "Narrative" Knowledge Representation Language (NKRL), see [12, 13] for more details. The aim is here that of suggesting that the use of NKRL in order to describe particularly complex "cases" can be of some benefit for the CBR community. NKRL — which, *inter alia*, has been recently used in two CEC-funded programs dealing with complex socioeconomic data, NOMOS ("*knowledge acquisition for NOrMative reasOning Systems*", ESPRIT P5330) and COBALT ("*COnstruction of knowledge BAses from natural Language documenTs in the financial domain* ", LRE P61011) — represents the last incarnation of a body of knowledge representation principles originally developed at CNRS (the French National Centre for Scientific Research) in the RESEDA project and its derivatives : these principles have definitely proved suitable for the representation of complex ("narrative") information.

### 2.1 The Four Components

In NKRL — which presents some rough similarities with the standard hybrid and terminological languages in the KL-ONE style, see, e.g., [3] — we make use of four neatly differentiated but interrelated components :

- The "descriptive component" concerns the representation of the semantic content of NL clauses describing some general classes of real-world events. In the context of the descriptive component, the

events taken into consideration must be structured events, i.e., characterized by the explicit indication of an actor, an object, an instrument, etc. Examples of such general classes may be "moving a physical object", "formulate a need", "having a negative attitude towards someone", "spreading academic knowledge", "be present somewhere", "come in possession of new resources", etc. The formal, NKRL representations of such general NL expressions are called "templates". Of course, for each template, its formal realization is independent of the surface structures of the possible, different NL utterances which can be used to describe the corresponding class of events.

- The "factual component" gives the formal representation of the NL narrative expressions relating some specific events — characterized, at least implicitly, by precise spatial and temporal coordinates — which constitute the concrete instantiations of the general class of the descriptive component. It concerns, therefore, the representation of NL expressions such as : "Tomorrow, I will move the wardrobe", "This morning, Lucy was looking for a taxi", "Mr. Smith has fired Mr. Brown", "Last year, he gave a course of lectures on Greek philosophy", "Peter lives in Paris", "Company X, located in Geneva, has taken the control of Company Y", etc. The NKRL expressions of these narrative statements take the name of "predicative occurrences". "Binding occurrences" are used to represent the logico-semantic links which can exist between the original events (e.g., a network of causal relationships).
- The "definitional component" concerns the formal representation of the main defining properties of all the general notions (at least partially proper to a specific application domain) which can be used in the framework of the descriptive and factual components. The corresponding NKRL data structures are called "concepts". Therefore, the definitional tools are used to represent in a concept format the essential properties of general entities like "*physical_object*", "*taxi_*" (the general class including all the taxis, not a specific cab), "*academic_knowledge*", "*resources_*", etc.
- The "enumerative component" concerns the formal representation of the instances (concrete examples) of the general notions (concepts) pertaining to the definitional component ; the NKRL formal representations of such instances take the name of "individuals". Therefore, individuals are created by instantiating (some of) the properties of the concepts of the definitional component. Individuals are characterized by the fact of being countable and of possessing unique conceptual labels ("smith_", "general_motors", "course_on_greek_philosophy_27") : two individuals associated with the same NKRL description but having different labels will be different individuals.

### 2.2 Representing temporal information

Particular attention has been paid, in a descriptive and factual components framework, to the efficient NKRL representation of temporal information. This possibility is of particular relevance in a CBR context, given that cases in most domains, and particularly in the socio-economic-political context, carry time-dependent information, and should therefore display some kind of ordering of (some of) their components — but in CBR practice this is most often built into the software to analyze the cases and is not explicit in the cases themselves. Exceptions to this last approach are described, e.g., in [4].

In the context of the representation of temporal information, NKRL's original contributions are, e.g., the notions of "category" and "perspective". The "category of dating" characterizes the association of a temporal marker to the beginning (the category is here the "posteriority", or "subsequence"), the end ("anteriority", or "precedence") or a particular moment ("contemporaneity", or "coincidence") of a given elementary event. The "perspective of dating" is used to define the degree of precision (for example, an incertitude expressed by a pair of dates) with which a given temporal marker is known. It can be shown that, *inter alia*, this formalism a) permits an integration of the "point" and the "interval" paradigms ; b) provides some tools (based on the concept of "perspective") in order to deal with the "fuzziness" which, in concrete situations, is often associated with the description of a temporal marker.

For a rough idea of the NKRL representation of temporal information, see, *infra*, Section 2.5 ; a recent paper on this subject is [14].

### 2.3 The NKRL data structures

The data structures supporting the four components are highly homogeneous, given that templates, occurrences, classes and individuals are all implemented as "structured objects" identified by an "OID" (object identifier). More precisely, the definitional and enumerative data structures are built up in a "frame" style. The structured elementary objects of the descriptive and factual components, on the other hand, are centered around a (unique) "semantic predicate" having its arguments introduced by means of "roles" as SUBJ(ect), OBJ(ect), SOURCE, etc. For example, the elementary occurrence (factual component) translating the NL sentence "John gives a book to Mary" will include a semantic predicate corresponding roughly to "coming in possession of something after a transfer" : "Mary" will be the SUBJ of "coming in possession", "book" the OBJ and "John" the SOURCE. The

descriptive and the definitional components are both implemented as *hierarchies* (or, more precisely, DAGs, Directed Acyclic Graphs) of structured objects.

### 2.4 The seven semantic predicates

According to what we said until now, the NKRL's structures are very general, and able to represent under the form of "cases" any sort of narrative context. Moreover, if this context is, as in the CEC-funded projects NOMOS and COBALT, a socio-economic-political one, where the main characters are human beings or social bodies, experience has shown that it is possible to make use, in the templates and occurrences of the descriptive and factual components, of only seven semantic predicates corresponding to very general, prototypical categories of human attitudes. They are described in Table 1.

| Predicate | Mnemonic Description |
|---|---|
| BEHAVE | A character adopts a particular attitude, or acts to obtain a particular result. |
| EXIST | To be present, also metaphorically, in a certain place. |
| EXPERIENCE | A character is affected by some sorts of good, bad or neutral news or events. |
| MOVE | The displacement of a person or a physical object, the transmission of a message ... |
| OWN | To have, to hold, to possess... |
| PRODUCE | Cause to exist or occur, with reference to material or immaterial entities, like the production of a service. |
| RECEIVE | To acquire, to obtain, without any connotation of mandatory or permanent possession |

Table 1. NKRL semantic predicates

In this case, thanks to the reduced number of basic semantic predicates, all the legal descriptive and factual structures and their practical modalities of use can be fully described in a "catalog" (see, e.g., the so-called "Stouder's catalog" [10]), thus allowing the use of these structures according to a coherent, reproducible and shareable strategy. Moreover, if necessary, new NKRL descriptive and factual structures can easily be derived from those already described in the "catalog".

### 2.5 An Example of NKRL Representation

As a very simple example of NKRL coding of a "case", let us consider the NKRL representation of the information : "Kurt Waldheim flies today to Baghdad in order to obtain from Saddam Hussein the release of the 95 Austrian hostages [25th of August, 1990]".

The coding will give rise to three occurrences (factual component), see Fig. 1 : two predicative occurrences, identified with the labels "a" and "b", and a binding occurrence, "c". This last occurrence is realized using GOAL, one of the four binding operators pertaining to the NKRL "taxonomy of causality", see, e.g., [13 : 705]. Occurrences "a" and "b" have two (mandatory) temporal determiners, the two date blocks "date-1" and "date-2" which are used to register the dating elements giving the limits of the temporal interval associated with the occurrence. In the case of the occurrence labelled as "a", only the first data block is filled because the situation described in this unit (Kurt Waldheim leaving Vienna on August 25th in order to meet Saddam Hussein in Baghdad) may be represented as a "point" on the time axis. In the occurrence "b", the two blocks are empty because, in the particular wording of the original piece of information (e.g., a news agency item about the 1990 Gulf crisis), the *actual* release of all the Austrian hostages is not stated expressly (the final result of the Kurt Waldheim's mission was still unknown when the notice has been issued). According to the original information, the situation represented by "b" must, therefore, be interpreted as "conjectural", i.e., it represents only, in a way, the "intentions" linked with the Waldheim's mission.

In "a" and "b" of Fig. 1, the arguments of the NKRL semantic predicates (entities of the domain) pertain to two categories : "Kurt_Waldheim", "Saddam_Hussein", "flight_7", "austrian_hostages_1" (the specific hostages of Austrian nationality in the "1990 Gulf crisis" context) are "individuals" (enumerative component) ; "*release_*" is a "concept" (definitional component hierarchy) which subsumes all sorts of *possible incarnations* of the general concept of setting a person free (we shall learn afterwards that the hostages have been handed over to Kurt Waldheim, and that they left Iraq using its own presidential jet). The argument introduced by the DEST(ination) role in "b" is a structured one ("expansion"), realized by using the quantifying attribute "95" inside a "SPECIF(ication)" list ; SPECIF is the "attributive operator". The colon code, ":", introduces the "location determiner" linked with a particular argument. In a MOVE construction, like "a", concerning the displacement of a character — in NKRL, we systematically represent this situation by expressing that the character, as a SUB(ject), moves himself as an OBJ(ect) — the location determiner (possibly, a list) associated with the SUB(ject) argument represents the initial location(s), and the determiner linked with the OBJ(ect) argument the final location(s).

---

| | | | |
|---|---|---|---|
| a) | MOVE : | SUBJ | Kurt_Waldheim : [ Wien_ ] |
| | | OBJ | Kurt_Waldheim : [ Baghdad_ ] |
| | | DEST | Saddam_Hussein : [ Baghdad_ ] |
| | | MODAL | flight_7 |
| | | [ date-1 : | 25_august_1990 ] |
| | | [ date-2 : | ] |
| b) | PRODUCE : | SUBJ | Saddam_Hussein : [ Baghdad_ ] |
| | | OBJ | *release_* |
| | | DEST | ( SPECIF austrian_hostages_1 95 ) : [ Iraq ] } |
| | | [ date-1 : | ] |
| | | [ date-2 : | ] |
| c) | ( GOAL a b ) | | |

Fig. 1. Predicative occurrences and binding occurrences.

---

## 3 The NKRL inference procedures

Strictly associated with the NKRL environment are different sorts of "standard" inference procedures. I will only evoke here the "transformations", i.e. a class of inference procedures which are proper to the descriptive and factual contexts, see, e.g., [11]. Transformations are declarative rules which allow a system organized around a conceptual knowledge representation language in the NKRL style to come up with information that was not exactly what was asked for, but nevertheless could be considered as a "plausible answer" to a given formal query. This result can be obtained by searching for *semantic affinities* between what is requested and what is really known by the system ; the fundamental principle adopted is then to *transform* the original query into one or more different queries, which are semantically close to the original one. Therefore, transformations allow us to implement an original form of *analogical reasoning*.

To give a very simple example, suppose that, working in the context of an hypothetical knowledge base about university professors, we should want to ask a question like : "Who has lived in the United States" even without an explicit representation of this fact in the base. If the knowledge base contains some information about the degrees obtained by the professors, we can tell the user that, although we do not explicitly know who lived in the States, we can nevertheless look for people having an American degree. This last piece of information, obtained by transformation of the original query, would indeed normally imply that some time was spent by the professors in the country, the United States, which issued their degree.

Without entering now in too many formal details, we can say that transformations are made up of a "left hand side" — formulation in a template (descriptive component) format of the linguistic expression which is to be transformed — and one or more "right hand sides" — representation in the same style of one or more linguistic expressions that must be substituted for the given one. A transformation can, therefore, be expressed as : " *A* (left hand side) → *B* (right hand side) ". The "transformation arrow", " → ", has a double meaning :

- an operational meaning, where the arrow indicates the *direction* of the transformation : the left hand side $A$ is dropped and replaced by the right hand side $B$ ;
- a logical meaning, where the arrow indicates that the information obtained through the use of $B$ *implies* the one obtained from $A$.

In reality, the "always true" implications (noted as " $B \Rightarrow A$ " in NKRL, where we assume that the symbol " $\Rightarrow$ " represents the "implication arrow") are not very frequent. Most transformations found in real world applications represent in fact "modalized implications" (noted as " B *$\Rightarrow$ A ", which means "it is *possible* that B implies A"). An example of this last type of transformations is given by the transformation t1 in Fig. 2, which will permit us to deal with the informal example above about "university professors" ; the left and hand right side of t1 are normal templates of the descriptive component, derived by basic templates described in the "catalog". Transformation t1 says : "If someone ( $x$ ) has obtained a title from an official authority by means of an official document, then it is possible that he has been physically present at that moment in the place ( $k$ ) where the authority is located".

This rule, for example, *is not always valid in* the case of an university degree (the degree could be obtained in a correspondence school, etc.). Nevertheless, it is easy to see that, in this case, the "semantic distance" between an "always true" implication and a "modalized" one is not too important, as it is always possible to change t1 into a true transformation by the addition of a few constraints on the variable $p$, for instance the "disequation" : "$p \neq$ < *obtainable_by_correspondence_degree* > ". Please note that all the constraints are realized by making use of concepts of the NKRL definitional component. More examples, and a complete semi-formal theory of transformations, can be found in [11].

---

t1) EXIST SUBJ $x$ : [ $k$ ] $\rightarrow$ OWN SUBJ $x$
OBJ $p$
SOURCE $q$ : [ $k$ ]
MODAL $r$

$x$ = < *human_being_* >
$p$ = < *title_* >
$q$ = < *authority_* >
$k$ = < *location_* >
$r$ = < *official_document* >

Fig. 2. A simple example of "transformation" rule.

---

In a CBR context, inference rules of the "transformation" type can demonstrate very useful for (at least) a) creating a very powerful unification module, based on some sort of "extended match operations", for comparing the (NKRL) description of a new case to that of the cases stored in a case base ; b) executing typical CBR operations, see [6], like the estimation of similarities or the adaptation of old cases.

## 4. Conclusions

We can conclude this short paper by mentioning another interesting aspect, in a CBR context, of an NKRL approach.

If we consider, in fact, the CBR indexing problem, the use of NKRL in order to represent cases should allow us to realize an up-to-date version of the principle of "indexing on complex features", see, e.g., [7, 8] — we prefer to use the term "conceptual indexing". "Conceptual indexing" goes back to R. Schank's original work on "memory organization packets" and "thematic organization packets" see, e.g., [9] and also [5] : this principle consists, very roughly, in the selection of some relevant characteristics of a specific type of representation and in the use of these characteristics to identify "semantic clusters". These clusters conceptually divide the knowledge bases into smaller modules, which are homogeneous from the point of view of the semantic content and which can be, at least partially, superposable.

According now to an NKRL approach, case bases where the cases are described by making use of the NKRL language can be indexed using as classification criteria some specific NKRL features, e.g. predicates, particular classes of concepts pertaining to the definitional component, the notions of "category" and "perspective" of dating, see, in this last context, the very detailed description given in [14]. Comparing these "main" characteristics with those appearing in the (NKRL) description of the case at hand, it is possible to "preselect" quickly a subset of the base containing all the cases which, according to the classification criteria adopted, are *a priori* likely to match the new case proposed. The true "match operations", which can be very complex, are thus

restrained to the match between the new case and the cases included in the reduced subset of the base created during the preselection phase.

## References


1. Alterman, R. (1989) "A Concept Space for Reasoninng About Cases Involving Event Structures", in: Proceedings of the 1989 DARPA Case-Based Reasoning Workshop. San Mateo (Calif.): Morgan Kaufmann.

2. Ashley, K.D., and Rissland, E.L. (1986) "Toward Modelling Legal Arguments", in: Automated Analysis of Legal Texts, Martino, A.A., and Socci Natali, F., eds. Amsterdam: North-Holland.

3. Brachman, R.J., McGuinness, D.L., Patel-Schneider, P.F., Resnick, L.A., and Borgida, A. (1991) "Living with CLASSIC : When and How to Use a KL-ONE-Like Language", in: Principles of Semantic Networks, Sowa, J.F., ed. San Mateo (CA): Morgan Kaufmann.

4. Campbell, J.A., and N. Chatterjee, N. (1991) Use of Time-Related Connectives in Representing Cases (Research Note). London: UCL Dept. of Computer Science.

5. Kolodner, J.L. (1984) Retrieval and Organizational Strategies in Conceptual Memory. Hillsdale (N.J.): Lawrence Erlbaum.

6. Kolodner, J.L. (1992) "An Introduction to Case-Based Reasoning", Artificial Intelligence Review, 6, 3-34.

7. Martin, C.E. (1989) "Indexing Using Complex Features", in: Proceedings of the 1989 DARPA Case-Based Reasoning Workshop. San Mateo (Calif.): Morgan Kaufmann.

8. Martin, C.E. (1989) Direct Memory Access Parsing (Ph.d. Thesis). New Haven (Conn.): Yale University.

9. Schank, R.C. (1982) Dynamic Memory : A Theory of Reminding and Learning in Computer and People. Cambridge: Cambridge University Press.

10. Stouder, L. (1987) RESEDA, le métalangage (Conv. INALCO/CIMSA SINTRA n° 0223A). Vélizy: Division CIMSA SINTRA de Thomson-CSF.

11. Zarri, G.P. (1986) "The Use of Inference Mechanisms to Improve the Retrieval Facilities from Large Relational Databases", in: Proceedings of the Ninth International ACM Conference on Research and Development in Information Retrieval, Rabitti, F., ed. New York: ACM.

12. Zarri, G.P. (1990) "A Knowledge Representation Language for Large Knowledge Bases and 'Intelligent' Information Retrieval Systems", Information Processing & Management, 26, 349-370.

13. Zarri, G.P. (1992) "The 'Descriptive' Component of a Hybrid Knowledge Representation Language", in: Semantic Networks in Artificial Intelligence, Lehmann, F., ed. Oxford: Pergamon Press.

14. Zarri, G.P. (1992) "Encoding the Temporal Characteristics of the Natural Language Descriptions of (Legal) Situations", in: Expert Systems in Law, Martino, A., ed. Amsterdam: Elsevier Science Publishers.

# Chapter 2

# Adaptation and Analogy

# An Analogical Reasoning Engine for Heuristic Knowledge Bases

*Jorge E. Caviedes*

Information Sciences Sector
Philips Laboratories
Briarcliff Manor, NY 10510

**Abstract.** One form of human analogical reasoning consists of solving problems based on their similarity to past episodes. In this paper we discuss a novel reasoning engine that performs analogical reasoning using traditionally engineered knowledge bases. This analogical reasoning engine learns inference rules that decompile heuristic associations from a knowledge base. Problems are solved by matching input data to plausible initial problem states and then applying learned rules, which decompile heuristics, to find a solution. An uncertainty management scheme has been developed that combines conceptual distance, probabilistic information, and rule uncertainty. The analogical reasoning engine emulates case-based reasoning using a simple architecture composed of one application-specific module and three domain-independent modules. The application-specific module is a heuristic knowledge base that uses a task model. The domain-independent modules are: a conceptual matching module, a rule-learning module based on a semantics-driven repertory grid analysis, and an uncertainty management module.

## 1 Introduction

Based on analogical problem solving, CBR systems rely on finding similarities to past cases to solve current problems [4,7]. Representation, indexing, and match and retrieval are among the issues contemplated in the design of CBR systems. In contrast with CBR systems, the purpose of heuristic KBs is to capture heuristic associations and solve problems following the approaches used by human experts. Heuristic KBs use well-known domain-specific or generic representation languages. Heuristic problem solving has at least three drawbacks which could be alleviated by adding analogical reasoning: it requires precise data matching, its overall consistency cannot be verified, and it involves intensive knowledge acquisition.

The analogical reasoning engine presented in this paper emulates case-based reasoning using a simple architecture composed of an application-specific module and three domain-independent modules. The application-specific module is a heuristic knowledge base that uses a task model; the domain independent modules are a conceptual matching algorithm, a rule-learning module based on repertory grid analysis, and an uncertainty management module. The analogical reasoning engine learns inference rules from the knowledge base, and solves problems by matching input data to plausible initial problem states and then applying the rules to find a solution.

This paper has the following outline: Section 2 deals with the components of the analogical reasoning engine, Section 3 deals with conceptual distance, Section 4 describes our approach to repertory grid analysis, Section 5 deals with uncertainty management, Section 6 presents the results of applying the analogical reasoning engine to the troubleshooting domain, and Section 7 contains the conclusions from this work.

## 2 The Analogical Reasoning Engine

The analogical reasoning engine is a problem solver that enables a heuristic KB to make use of analogical reasoning. The four main modules shown in Figure 1 are: an application-specific KB and task model, a conceptual matching module, a rule learning module, and an uncertainty management module. We have developed two new algorithms, one for conceptual matching and one for uncertainty management. For the rule learning module we have used a technique called repertory grid analysis, with a new semantic element added to it. The roles of the four modules are described below, and the details of the algorithms will be given in the following sections.

**Application-specific module.** This module can be any type of KB that contains heuristics and an abstraction of the problem solving approach called the task model. A task model is a notion derived from decision-making models and human-performance models. It simply captures the decision making stages or main subgoals of problem solving used by human experts (i.e. troubleshooting has three subgoals: diagnose, repair, and verify).

**Conceptual Matching Module.** The purpose of this module is to match the input problem to plausible initial problem states. The analogical reasoning process starts when a problem description is entered as free form text. The conceptual matching algorithm identifies keywords in the text and looks for matching strings in the KB. In

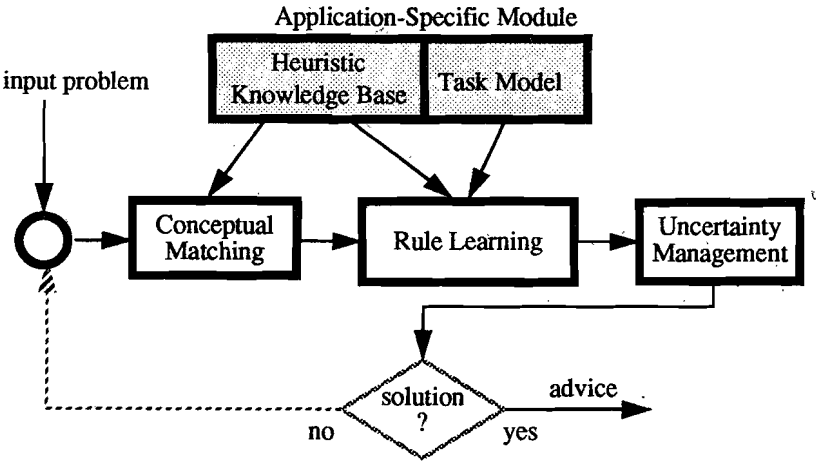



Figure 1. Analogical Reasoning Engine

this fashion, domain-specific concepts such as symptoms, functional units, or pathological states can be identified and assigned a conceptual distance to the problem input. The conceptual distance measure allows the generation of an ordered set of plausible initial problem states. The analogical reasoning engine, however, does not take the approach of simply retrieving the heuristic answer to the most likely problem state.

**Rule Learning Module.** This module learns rules underlying heuristic associations. A measure of uncertainty is associated with the premises and the conclusions of the rules. The analogical inference engine finds an answer by applying these heuristics-decompiling rules to the conceptually matching data. Decompiling a heuristic association means making explicit at least one intermediate conclusion used by that heuristic without engaging in the type of model-based reasoning described in [2]. Which and how many types of heuristics should be decompiled varies depending on the task model. Usually, there is one key heuristic that applies to all problems and decompiling it is enough. Otherwise more than one heuristic should be decompiled. At least the main heuristic should be decompiled so that the analogical reasoning engine may be able to generate more precise advice than the KB, or generate advice which was not apparently available before.

**Uncertainty Management Module.** This module does the computation of accumulated uncertainty from the rules and the conceptual matching. The goal of the uncertainty management module is to fine tune the matching of problem data to KB knowledge. If a solution is not found at the end of one cycle of matching and inferencing, the process can be repeated as desired; otherwise the system would give up. The uncertainty management module makes sure that uncertainty values from the conceptual matching and the rules, plus any available probabilistic data are accumulated and carried through to the final conclusions. If any solutions are found, they are ranked by their overall certainty.

If no solutions were found in the first pass, under an open loop implementation the system would give up. Under a supervised closed loop implementation, a new cycle that uses a lower uncertainty threshold starts, and recomputes the initial matching states, with or without further interaction with the user. Data from unsolved cases from both open and closed loop configurations should be used to refine the knowledge base off line, using tools available in the original KB development environment.

## 3 Matching Textual Descriptions and Conceptual Distance

One of the mechanisms to select relevant knowledge base objects is conceptual matching between the problem input in free text form and textual information in the KB which is linked to domain objects. Syntactic matching between two textual strings $S_1,S_2$ is done using a metric $d(S_1,S_2)$. We have developed a conceptual distance measure which is based on the similarity among keywords related to domain concepts found in two text strings. The conceptual distance uses the matching score between two keywords $KW_1$ and $KW_2$ defined as:

$$KW_{score}(KW_1, KW_2) = \frac{M_{chars} + O_{chars} - |L_1 - L_2|}{L_1} \leq 1$$

$M_{chars}$ is the number of identical characters from $KW_1$ which are found in the same position in $KW_2$, and $O_{chars}$ accounts for the remaining identical but shifted characters as follows: 0.9 if the character is one position away, 0.8

if the character is two positions away, and 0.5 if the character is more than two positions away. Based on this non-symmetrical matching score, the distance between two keywords has been defined as:

$$d(KW_1, KW_2) = 1 - KW_{score}(KW_1, KW_2)$$

The matching of text strings, $S_1$ and $S_2$ with $n_1$ and $n_2$ words respectively, is analogous to the matching just described for keywords. The maximum scores for each keyword are accumulated and normalized by the number of keywords in the target, minus a weighted penalty for length difference. The formulae are the following:

$$StoS_{score}(S_1, S_2) = \frac{\sum_{i=1}^{n_1} \max\{KW_{score}(KW_i, KW_k), k \in [1,n_2]\}}{n_1} - LDF * \frac{|n_1-n_2|}{n_1}$$

$$StoS_{distance}(S_1, S_2) = 1 - StoS_{score}(S_1, S_2)$$

LDF is the *length difference factor*. Empirical observations indicate that totally unrelated strings usually lie at a conceptual distance greater than 0.65 when the LDF is set to 0.3. Thus, we have used 0.65 as the cutoff for conceptual distance.

## 4 An Algorithm for Rule Learning: Semantics-Driven Repertory Grid Analysis

Repertory grid analysis is a technique which has been applied with some success to the automation of knowledge acquisition [1,3]. It is based on the personal construct theory, which holds that people evaluate their own experiences by means of bipolar distinctions called constructs, which apply to items called elements. For instance, the construct "driven/unmotivated" may be used by teachers to evaluate their students as follows: the students are assigned integers from 1 to 5 to express their rating within the two extremes of the construct (e.g. extremely driven 1, extremely unmotivated 5, neutral 3).

| | | SYMPTOMS | |
|---|---|---|---|
| | no prints no errors | • • • | |
| TESTS: test power cord | 1 | *extremely necessary = 1*<br>*extremely unnecessary = 5*<br>*neutral = 3* | ▲▼ |
| TESTS: test power switch | 2 | | |
| TESTS: test power input | 3 | | |
| COMPONENTS: power supply | 1 | *extremely involved = 1*<br>*extremely not involved = 5*<br>*neutral = 3* | ▲▼ |
| COMPONENTS: interface | 3 | | |
| COMPONENTS: paper feed | 3 | | |

Figure 2. Repertory Grid Analysis for Troubleshooting Heuristics

From analysis of the grid ratings it is possible to generate classificatory rules which may be applied to elements outside the original set. Let us consider an example for the checklist heuristic from the troubleshooting domain. The elements are the symptoms, and the constructs are tests and components (i.e. the necessity to do the tests and the involvement of the suspect components). In general, the elements must be the inputs to a key heuristic, and the constructs are at least two types of outputs of the heuristic (i.e. one of them is the type of an intermediate conclusion of the heuristic). A repertory grid like the one shown in Figure 2 has been used to decompile checklist heuristics.

We have introduced the notion of semantic categorization of the constructs, suspect components and diagnostic tests, which leads to distinct grid regions that determine the rule types that can be generated (see the regions on the cross reference matrix in Figure 3). With the semantic categorization, the analogical reasoning engine can insure that only rules which can be interpreted in the light of the task model are generated, thus avoiding the problem of rule interpretation.

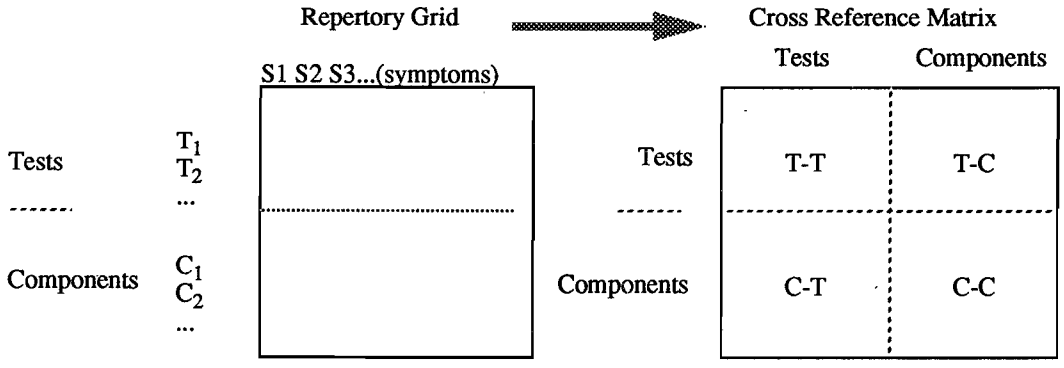



Figure 3. Semantic Categorization and Types of Rules Generated from Grid Data

Four types of rules can be learned: T-T (test to test), T-C (test to component), C-T (component to test), and C-C (component to component). For each element of the cross reference matrix it is possible to generate more than one rule. In general, the rules have the following format:

$[Q_i \text{ CONSTRUCT}_i \textit{ entails } Q_j \text{ CONSTRUCT}_j]_{CF}, i \neq j$

This means that any two different constructs could be linked by a rule with a confirmation factor computed from the evidence available on the grid. The qualifiers $Q_i$ and $Q_j$ are the qualitative interpretations of the ratings and can be any single value or a meaningful range, for example: *extremely needed (1), at least somewhat needed ($\leq$2), extremely involved (1)*, etc.

A $C_i$-$T_j$ rule would clearly indicate how necessary the diagnostic test $T_j$ is when component $C_i$ is a suspect. To generate such a rule, with qualifiers *extremely involved* and *extremely needed* (grid ratings = 1), from a grid with 10 symptoms we would need the following grid rows:

$C_i$ data, or the i-th row in the Repertory Grid: 1 3 5 2 1 3 1 3 3 3

$T_j$ data, or the j-th row in the Repertory Grid: 3 3 5 3 1 3 3 3 3 3

The rating values of 1 are defined as being on the alpha plane 1 or $\alpha_1$. The confirmation factor for the rule is the ratio between the positive relevant evidence and all relevant evidence, as shown in Figure 4. (This factor is really the amount of partial entailment of $T_j$ given $C_i$.) Notice that all relevant evidence is located on the alpha plane $\alpha_1$ and all information for $C_i$ which is not on $\alpha_1$ is irrelevant evidence.

| Relevant Evidence | |
|---|---|
| Positive $C_i\&T_j@\alpha_1$ | Negative $C_i@\alpha_1\&T_j@\sim\alpha_1$ |
| 1 | 2 |
| Total: 3 | |

$$CF = \frac{\text{Positive Rel. Ev.}}{\text{Total Rel. Ev.}} = \frac{1}{3} = 0.33$$

all information for $C_i@\sim\alpha_1$ is irrelevant

Figure 4. Computation of the Confirmation Factor for Rule $C_i$- $T_j$

Our use of repertory grid analysis has the following innovations: (i) semantic categorization of the constructs, (ii) selective learning, so that only rules relevant to the task model are generated and invoked, and (iii) rule application following a task model and using analogical reasoning.

## 5 Uncertainty Management

Several types of uncertainty accumulate along the process of solving a problem with the analogical reasoning engine. The conceptual distance, the uncertainty of the rules, and any probabilistic information available (e.g. known failure probabilities) need to be combined to generate a certainty measure for the final conclusion. For the troubleshooting example we developed the formula shown in Figure 5. Due to its linearity and simplicity, this

scheme can be generalized as the product of the conceptual distance to the closest initial problem state, the a priori probabilities of the preconditions of relevant rules, and the certainty factors of the conclusions of the rules.

Formula to rank recommended tests:

T_rts = [ 1 - S_d ] * C_fp * T_cf

S_d Symptom Conceptual Distance

C_fp Component fault probability

T_cf C-T rule confirmation factor

Figure 5. Uncertainty Computation Formula

## 6 Results

We applied the analogical reasoning engine to a troubleshooting KB built using DSS [5]. The KB was designed for the troubleshooting of laser printers. As shown in Figure 6, analogical reasoning helped to generate diagnostic tests for complaints stated in plain English by (i) matching the input against symptom descriptions, (ii) from the likely symptoms, one set of diagnostic tests and another set of suspect components were generated, (iii) C-T rules were learned and used to infer recommended tests, and (iv) the recommended tests were ranked based on overall certainty.

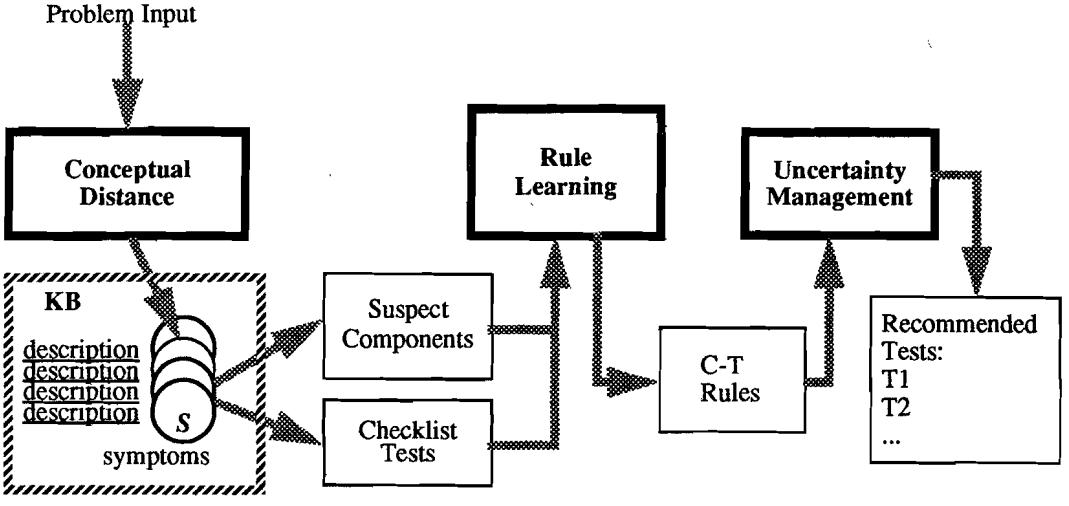



Figure 6. Analogical Reasoning for a Troubleshooting KB

The elements of the repertory grid were symptoms, and the constructs were checklist tests, and suspect components. The main heuristic which was decompiled was the association between symptoms and diagnostic tests called the *checklist* [5]. The grid ratings used were simply a 1, or *strongly needed* or *involved*, if the test or component was present in the checklist or in the suspects list, and 3, or *neutral*, otherwise. The checklist is thus decompiled as the sequence "first generate the suspect components, and then find the optimum test for the most likely faulty component".

We performed two experiments done using different types of input. In the first experiment we entered verbatim the known symptom description "prints have a wavy pattern" to analyze the extent of agreement with its checklist in the KB. In the second experiment we entered a more ambiguous description.

As expected, in the first case the conceptual distance to the target symptom was 0.0, and the recommended tests included the original checklist items with the highest scores. Other recommended tests came from the checklist of a symptom which lies at a conceptual distance of 0.5625. In the second experiment, we entered an ambiguous and misspelled description of a known symptom. The results showed that the conceptual distance matching works well, selecting the right symptom, and the only member of the original checklist is one of the recommended tests. The other recommended tests were found to lead to plausible diagnoses according to the original KB.

All learned rules must be validated by a domain expert before they are used in an application. Otherwise, unexpected conclusions may need to be analyzed in order to judge the value of the rules. In a few cases, tests which

were not in the original checklist were highly recommended. Upon closer examination of the knowledge base, we found that in one case the new test corresponded to the checklist of a sub-symptom of the current symptom, and the evidence for the use of that test was stronger than the evidence for any of the tests in the original checklist (it had highest confirmation factor for the C-T rule). Thus, we concluded that given the data available it could have been a more effective test for diagnosing the suspect component.

## 7 Conclusions

We have presented an innovative approach to analogical reasoning for heuristic knowledge bases which takes advantage of existing KB architectures. The analogical reasoning engine adds flexibility to heuristic problem solving by broadening the range of input data. With this approach we have shown that the specialized case-based architecture can be emulated by the analogical reasoning engine, which only requires a heuristic knowledge base and its task model.

The analogical reasoning engine has three domain-independent modules which include the following specific innovations: a conceptual matching algorithm, a semantics-driven repertory grid analysis tool, and an uncertainty management scheme. The semantics-driven repertory grid analysis has the advantage of solving the problem of rule interpretation since the semantics of the rules generated are given by the grid region on which they fall. The grid can also be used to build new KBs by letting the user create rows for categorized constructs, and columns for elements whose relationships with the rows can be rated by the values entered in the grid. This spreadsheet-like approach would enable new consistency tests based on cross referencing, which may help the KB designer validate the notions of structure and behavior implicit in heuristic associations. When applied to existing KBs, as in our example, the constructs and elements for rows and columns already exist and are simply placed on the grid and the ratings are filled in with default values such as extremely relevant or neutral, corresponding to whether two concepts are related or not.

The improved utilization of heuristic knowledge bases has broad effects. Knowledge bases are usually designed for one specific delivery environment, and the knowledge is intentionally entered to be used in one manner. However, analogical reasoning provides the flexibility of using the knowledge base in a different mode. In the case of troubleshooting, heuristic knowledge bases are built for field assistance. With analogical reasoning, help desk consultations are possible given that one can enter loosely stated problems, just as they are stated by non-technically oriented customers who want the best advice possible.

**References**


[1] J.H. Boose, A Knowledge Acquisition Program for Expert Systems Based on Personal Construct Psychology, *Intl. J. Man-Machine Studies*, Vol. 23, 1985, 495-525.

[2] R. Davis, Diagnostic Reasoning Based on Structure and Behavior, *Artificial Intelligence*, Vol. 24, 1984, 347-410.

[3] K.M. Ford, F.E. Petry, J.R. Adams-Webber, and P.J. Chang, An Approach to Knowledge Acquisition Based on the Structure of Personal Construct Systems, *IEEE Transactions on Knowledge and Data Engineering*, Vol. 3, No. 1, March 1991, 78-88.

[4] J.L. Kolodner, Improving Human Decision Making through Case-Based Decision Aiding, *AI Magazine* Vol. 12, No. 2, Summer 1991, 52-68.

[5] M.K. Reed, J.E. Caviedes, and T.S. Anand, A Domain-Specific Environment for Developing Expert Systems, *Expert Systems: Planning/Implementation/Integration*, Vol. 2, Winter 1991. Auerbach Publishers, NY, 1991, 15-25.

[6] Simoudis E., Using Case-Based Retrieval for Customer Technical Support, *IEEE Expert, Intelligent Systems and their Applications*, Vol. 7, No. 5, October 1992, 7-12.

[7] S. Slade, Case-Based Reasoning: A Research Paradigm, *AI Magazine*, Vol. 12, No. 1, Spring 1991, 42-55.

# Adaptation Through Interpolation for Time-Critical Case-Based Reasoning

N.Chatterjee J.A.Campbell

Department of Computer Science
University College London
London WC1E 6BT
U.K.
email: {nchatter,jac}@uk.ac.ucl.cs

## Abstract

The paper introduces and examines the relevance of the notion of "interpolation" between case features, to facilitate fast adaptation of existing cases to a current situation. When this situation is time-critical there is not enough time for exhaustive comparison of various aspects of all the stored cases, so it may not be possible to retrieve a high-quality match for a current problem within a specified time-limit. Viewing imperfect adaptation as a process of interpolation (or a set of possible processes with different qualities of interpolation) then gives the best and most robust perspective for time-critical reasoning. Although interpolation-like adaptation techniques have been used in some existing CBR systems, they have not previously been treated explicitly from this perspective.

## 1. Introduction

*Interpolation* is a well-known technique for quick solution of numerical problems. In this paper we extend the numerical-interpolation idea to symbolic values in order to achieve efficient adaptation for Case-Based Reasoning (CBR) systems. Efficient adaptation is one of the prime requirements for CBR systems in general. But in systems dealing with time-critical problems this need becomes much more acute as the performance here can be characterised primarily by features like speed and timeliness i.e. ability to finish a task within a stipulated time [6]. Since an upper limit of allowable time to find a solution is prescribed, a CBR system solving time-critical problems usually cannot afford the standard methods of wading through an entire case-base, judging each case individually, in order to retrieve the best possible match. Consequently the system is left with the option of modifying the best possible past case(s) that it could retrieve within an allotted time period - or in extreme situations may have to adapt some 'default' solutions stored against such contingency [3]. Naturally, an effective general adaptation approach is required for a CBR system working in a time-critical environment. We offer the notion of interpolation as a means of fulfilling the need for a domain-independent, quick and efficient adaptation tactic.

In some of our earlier work [4] we proposed the notion of interpolation for knowledge-based systems, and argued briefly that its special requirements could make it even more suitable to apply to cases than to other knowledge representation schemes. In this paper we explain how 'interpolation' can be extended to cover symbolic values and examine in detail the kinds of interpolation that can occur.

However, we feel that 'interpolation' is not an entirely new concept in case adaptation. As we review critically the papers on existing CBR systems, we find that in many of them some ad hoc steps have been taken which closely resemble some of the interpolation techniques we suggest below. This observation not only authenticates the suitability of interpolation methods for adaptation, but also paves the way for an in-depth look at it. This paper in that sense is a pioneering effort to put interpolation in CBR on a uniform footing.

## 2. Interpolation in CBR - the Approach

The basic motivation for interpolation comes from numerical analysis, where it is often used as "the technique of approximating a function in order to evaluate it at some unknown point when the values are known for a set of tabular points" [5]. And one uses the term "extrapolation" for the same general procedure when the unknown value is not too far off one end of an interval whose properties are known.

This paradigm is obviously attractive for case adaptation provided that it can be made to have a good degree of uniformity (domain-independence) and applicability outside the range suggested by the example of numerical

analysis. However, extension of the idea of interpolation to non-numerical problems is not straightforward - which is probably why 'interpolation' has so far remained uncited in knowledge-based problems. Hence some careful observations are needed for making the basic tactics of interpolation clear.

The most immediate property of numerical values is that they have an inherent order. Therefore when a new quantity is encountered it is simple to determine its relative position with respect to other quantities, making interpolation straightforward to apply. But that is usually not so for features in realistic knowledge-intensive problems, where symbols are used predominantly. As no well-defined order is immediately present between symbols, interpolation in symbolic domains requires some means of imposing an order (or a partial order, at the least) on the features.

Now, an order between two symbols is meaningless unless an attribute that is relevant in the current context is considered. For example, there is no obvious order on the animals deer, cow and elephant. But when we talk about any specific property we can hope to find some order with respect to it: cow falls between deer and elephant when the attribute is weight, when speed is important elephant comes between cow and deer, while deer falls between elephant and cow when potential for domestication is the attribute to consider. A metric can be set up in any one dimension to assign relative distances between pairs of entities, and distance in multiple dimensions (multiple properties) can then be calculated via standard metrics (Euclidean, Manhattan etc.). We use the word 'interpolation' to cover both interpolation and extrapolation, in the sense mentioned above, with respect to knowledge-based problems as well.

In the next section we indicate how different types of metrics can be specified for symbolic quantities.

## 3. Different Ways of Imposing Metric Interpretations

In connection with our CBR work we have identified 8 different ways to impose metric interpretations on symbolic quantities. We illustrate them through real-life examples or examples from existing CBR systems where they are already expressed in a similar language.

Most straightforward is the situation when the feature itself has implicit order. This situation has two subdivisions.

### 3.1. Numerical Values

There are certain features which can be characterised adequately by numerical values only, e.g. distance, time, weight. Here people often use straightforward interpolation (rather unconsciously and subjectively, perhaps). For example: when A asks B how much it should cost to go to Victoria from King's Cross by taxi, B immediately replies that it should be around 7 to 8 pounds, from his tentative idea that the trip should take around 20 minutes by road and last time when he travelled by taxi he paid 2.50 pounds to go from Oxford Street to Euston station which took nearly 6 minutes of driving.

### 3.2. Symbolic Quantities Masking Numerical Values

There are features which are measurable according to some standard scale (which may be known only to the domain experts and not the people who record the raw case data), yet in non-expert practice are expressed in symbolic terms. The common choice of an ordering of colours according to the wavelengths of the corresponding light or use of musical notes instead of their frequencies are simple relevant examples here.

For features that are not directly metrisable we suggest two different types of artificial metric: artificial enumeration through ordinals, and fuzzy.

### 3.3. Fuzzy Quantifiers

Here we can borrow from the standard techniques of fuzzy reasoning, e.g. translating fuzzy terms into distributions, performing convolutions to derive distributions expressing combinations of terms, and making inverse translations to find the right fuzzy quantifier for the result. The formality of the treatment distinguishes this scheme from the one immediately above, even though their terms may overlap. Zadeh [10] discusses this in detail. In real life people often use fuzzy terms e.g. very-big/ big/ small/ fairly-small (for size), heavy/ medium/ light (for weight), very high/ high/ low (for temperature) etc. instead of actual quantitative values. Similar concepts can be seen in the system for fault recovery in automated machinery designed by Barletta and Mark [1]. When they use explanations like 'since the current temparature is 110 degrees, the viscosity is low and

therefore manual rotation is not needed', they are basically applying interpolation (according to our definition) on domain knowledge like 'if the lubricant is too cold then viscosity is too high'.

### 3.4. Artificial Enumeration Through Ordinals

Often artificial orderings are applied to symbols to express their relative order e.g. for educational qualifications of persons, postgraduate 5/ graduate 4/ diploma 3/ A-level 2 etc. CHEF [7] uses this technique for describing object properties. For example, knowledge of the form 'taste of broccoli is savoury with intensity 5.' and 'taste of beef is savoury with intensity 9.' corroborate this artificial enumeration. Whatever consequent recipe-adapting actions CHEF takes (on the basis of these values) in order to replace one ingredient by another in some dish falls within the scope of this heading.

It should also be noted that the ordinals are normally set with respect to certain particular feature. Thus a post-graduate may get ordinal number 2 while a diploma-holder may get 5 when the relevant feature is 'ability to do electrical repairing works'.

Other interpolation methods that we have used are:

### 3.5. Discrete Selection: choosing one of a finite set of alternatives

Examples of this can often be found implicitly in real life, although they are usually not described in terms similar to the heading above. For example, it may be necessary for a maintenance worker to fit into a confined space (which implies a suitable small size and high level of agility) and to be strong enough to manipulate heavy objects in that space (which puts premium on strength and therefore downgrades small size). A foreman may select X from the list of available workers because X is a member between Y who is too weak and Z who is rather too large for the job, given also that there is some positive information about agility on X's record.

### 3.6. Optimisation of Certain Functions

Often compromise between two conflicting demands is arrived at by providing an alternative that maximises the the total satisfaction of the two parties involved. For example, Mr. X wants to spend the holidays in the mountains of Austria while Mrs. X, who does not like hills, prefers to go to a French seaside location. Mr. X being totally reluctant to go to a sea resort, the ultimate solution they arrive at is to go to a historical place like Rome. Although it may be 2nd and 3rd in the preference lists of Mr. and Mrs. X, it maximises their joint utility function. Interpolation involves finding an intermediate point by applying a guiding criterion, (i.e. a search for an extremum) here.

Examples of this type of interpolation can be found in JULIA, the case-based meal planner [8]. For example, the conflicting goals of a host, who wants to serve egg as the main ingredient, and an invitee, who is on a low-cholesterol diet, is solved by the planner by suggesting a menu with egg as the secondary ingredient.

### 3.7. Rule Sets

To keep close to our paradigm of interpolation, the postconditions of the rules must be of the form "given x, y, z, f(x) and f(z), where $x < y < z$, derive f(y) in the following ... way". The method of derivation indicated by ... will vary from rule to rule: a rule's preconditions will therefore determine a context. Where 'f' gives a ratio between amounts of different types of financial instrument, in the portfolio of a stockbroker's client, and the context is a financial outlook described by national quantitative economic indicators, expert systems produced in the EQUUS project [2] perform just such an interpolation.

### 3.8. Iterative Interpolation

Above, we have mentioned an interpolation that is a compromise between two values A and B. Applying a guiding criterion, as mentioned there, is in general likely to involve an iterative improvement on some rough initial solution. Here, each of A and B has associated with it a set of tests or conditions which serve as justifications for a value determined by some more elementary type of interpolation to be moved closer to itself. When neither set can offer a justification stronger than some predefined threshold, the interpolation terminates. The resulting picture is of a bargaining and counter-bargaining dialogue. PERSUADER, a system to imitate mediation in labour disputes [9], applies this kind of interpolation.

## 4. An Example

In this section we illustrate a use of interpolation with an example from our current CBR problem domain of ground operations control of an airport. Suppose the current situation is as follows:

> A plane which is due to depart in 15 minutes has a problem with a piece of defective radio equipment, which can be repaired and recalibrated in place only after a very long delay, or within 15 minutes if the plane is (re)located close to a concourse window from which connections to electronic facilities can be made. The situation is time-limited because an incoming flight is scheduled to unload at the gate occupied by the original plane in 20 minutes.

No case is found in the case-base with exactly the attributes above. The most similar cases differ basically from the current one in a "dimension" that we can summarise, for the purposes of this paper, as "difficulty-in-repair-or replacement". The two closest cases are (i) one in which a plane is delayed by a defective baggage-door, (ii) one in which a crack has been found in an external window, necessitating replacement of this window. Common to both cases are the instructions

- call engineers to repair the defect (with different expected times for the repair, e.g. 5 minutes for the window and 15 for the door);
- ensure that the repaired plane will leave as soon as the work is completed;
- make a path plan for the incoming plane.

The plan in the "window" case also contains the instruction "ensure that a special component" (the replacement window) "is ordered", while there is no such instruction in the "door" plan. In this instance the interpolation must select one or the other (discrete selection, stated above); there is no intermediate possibility. The controller is therefore prompted to ask the most accessible specialist (e.g. the crew member in charge of navigation etc.) if it is desirable to order a special component - while the absence of the interpolation scheme might have meant that such an order could have been overlooked until far too late.

Also relevant to interpolation (of different operations on the same data) is the fact that the "door" case contains an instruction replan-flight-movements with one argument indicating the time horizon over which readjustments are likely to be necessary, while in the corresponding position in the "window" case there is an instruction readjust-identity, with the same type of argument. (This refers to a situation in which the window defect was in a plane allocated to a flight over a significant distance, while an identical plane allocated by the same airline to a short-haul flight was at an adjacent gate). In this environment there is a set of similar operations differing in their ranges of effect in the airport. The set includes a readjust operation, available for any airline that occupies a sequence of gates, whose effect is to ask the manager for that airline's operations to consider some rearrangement, of any type (relabeling, physical movement), for planes assigned to those gates. Here, this suggests the possibility of moving the plane with the defective equipment to a gate adjacent to a concourse. The interpolation selects the readjust operation.

In a final simple interpolation, the expected repair time is estimated as 10 minutes (interpolating numerically between 5 minutes for window and 15 minutes for door).

## 5. Concluding Remarks

In the full-sized paper we intend to illustrate the effectiveness of 'interpolation' as a rapid adaptation technique, via examples from our time-critical problem domain of ground operations control of an airport, using instances of all the types of interpolation mentioned above, and indicating how to choose an appropriate type automatically for each problem. Our work has shown further that (with the help of a caching scheme [3]) results can be computed reliably within the time limits set by the time-critical requests. An essential feature of our treatment of time-criticality is that different interpolation schemes, each with a time cost determined by past experiments, are available, and the best interpolation consistent with the time allowed is chosen automatically [3].

While the main emphasis of this abstract is on interpolation, a full justification for interpolation in use relies on being able to express relative distances between symbolic terms. As indicated above, this is a standard activity in numerical taxonomy, but notions of semantic distance are also occasionally used in AI. Discussion of this issue can be expanded within a full-sized paper, if required.

## 6. Reference

[1] Barletta R. and Mark W. : Explanation-based Indexing of Cases. Proceedings of 7-th National Conference on Artificial Intelligence. Saint Paul, Minnesota August 21-26, 1988, pp 541-546 .

[2] Campbell J.A. : Assessing the Quality of Knowledge-Based Systems: Lessons from the EQUUS Project. Heuristics 3, 1991, pp 9-17.

[3] Chatterjee N. and Campbell J.A. : A Caching Scheme for Time-Critical and Case-Based Reasoning. Proc. EXPERSYS-92, IITT-international, F-93460 Gournay sur Marne, France, pp 477-482.

[4] Chatterjee N. and Campbell J.A. : A Caching Scheme for Time-Critical Knowledge-Based Computations. Accepted for Sixth International Conference on Industrial and Engineering Applications of Artificial Intelligence and Expert Systems. Edinburgh, June 1-4, 1993.

[5] Conte S.D. and de Boor C. : Elementary Numerical Analysis : An Algorithmic Approach , McGraw Hill, 1972.

[6] Dodhiawala R., Sridharan N.S., Raulefs P. and Pickering C. : Real-Time AI Systems : A Definition and An Architecture, Proc. IJCAI-89, Morgan Kaufmann, San Mateo, California, 1989, pp 256-261.

[7] Hammond K.J. : Explaining and Repairing Plans that Fail. IJCAI-87, Milan, Italy, August 23-28, 1987, pp 109-114.

[8] Hinrichs T.R. and Kolodner J.L. : The Roles of Adaptation in Case-Based Design. Proc. DARPA Case-Based Reasoning Workshop, Morgan Kaufmann 1991, pp 121-132.

[9] Sycara K. : Using Case-Based Reasoning for Plan Adaptation and Repair. Proceedings Case-Based Reasoning Workshop DARPA, May 10-13, 1988 , Clearwater beach, Florida, pp 425-434.

[10] Zadeh L.A. : A Theory in Approximate Reasoning in Machine Intelligence 9, eds J.E. Hayes, D. Michie and L.I. Mikulich. New York, Halstead Press pp 149-194, 1979.

# Knowledge Engineering Requirements in Derivational Analogy

Pádraig Cunningham, Seán Slattery

Department of Computer Science, Trinity College Dublin, Ireland

Donal Finn

Hitachi Dublin Laboratory, Trinity College Dublin, Ireland

**Abstract.** A major advantage in using a case-based approach to developing knowledge-based systems is that it can be applied to problems where a strong domain theory may be difficult to determine. However the development of case-based reasoning (CBR) systems that set out to support a sophisticated case adaptation process does require a strong domain model. The Derivational Analogy (DA) approach to CBR is a case in point. In DA the case representation contains a trace of the reasoning process involved in producing the solution for that case. In the adaptation process this reasoning trace is reinstantiated in the context of the new target case; this requires a strong domain model. In this paper we analyse this issue using as an example a CBR system called CoBRA that assists with the modelling tasks in numerical simulation. We conclude that CBR systems for more innovative tasks should focus on interactive adaptation.

## 1 Introduction

Case-Based Reasoning (CBR) has emerged from research in cognitive psychology as a model of human memory and reminding. It has been embraced by researchers on AI applications as a methodology that avoids some of the knowledge acquisition and reasoning problems that occur with other methods for developing knowledge-based systems. One of the central advantages in using a case-based approach to developing knowledge-based systems (KBS) is that CBR systems can be developed without encoding a strong domain theory for the problem domain [1]. However there are CBR systems that incorporate a strong domain theory. Systems that set out to support a sophisticated case adaptation process do require a strong domain model. So there is some question as to whether these CBR systems with deep knowledge representations loose this central advantage of the CBR approach to KBS development.*

The Derivational Analogy (DA) approach to CBR is a case in point [2][3][4]. In DA the case representation contains a trace of the reasoning process involved in producing the solution for that case. In the adaptation process this reasoning trace is reinstantiated in the context of the new target case. If the domain model is to support the reinstantiation of a reasoning trace then it will have to be a fairly comprehensive representation.

In this paper we will attempt to analyse this issue using as an example a CBR system called CoBRA (Case-Based Reasoning Assistant) that assists with the mathematical modelling tasks in numerical simulation. CoBRA is a DA based CBR system that produces simplified models of cooling fins for heat and fluid flow analysis. CoBRA's cases consist of model descriptions and a trace of the model simplification process (see Fig. 3). The adaptation process attempts to reapply this reasoning trace to the fin model in the target case. CoBRA contains a fairly sophisticated domain model to support this adaptation. The question is: is the advantage of CBR lost in having to support it with a deep model of the problem domain? Could this system have been developed as readily by encoding the knowledge as a planning system of transformation heuristics represented as rules?

Before looking more closely at CoBRA we will look at a simple CBR system called Rachmann for estimating house prices. This system will act as a touch stone to mark the simplicity of the basic CBR process.

## 2 Rachmann: A classic CBR system

Rachmann is a small CBR system for property valuation. Each case is a property represented as set of features and the value of that property (see Fig. 1). A target case is a set of features representing a property for which a valuation is sought. The systems finds the best match from its case base and performs simple adaptations on that case to determine a valuation for the target case.

The advantages of CBR for knowledge acquisition are manifest in this example. The cases are easy to set up as the features are obvious important attributes of houses affecting the market value. However, the system is not completely without a domain theory because the organisation of the indices in the discrimination network reflects their relative importance. In addition, the partitioning of a city into locations reflecting property values

* It is worth mentioning that these comments apply to the use of CBR in developing KBS and not to the use of CBR as a model of memory and mental processes.

requires some expertise. Nevertheless, our experience has been that this system is much easier to set up than an equivalent system using heuristics to reason from first principles.

| 4WF | *Indices* |
|---|---|
| Location: | SM-1 |
| B-Rooms: | 2 |
| Age: | Modern |
| Rec-Rooms: | 1 |
| Kitchen: | Small |
| Rear-Acc.: | No |
| Tot-Area: | <800 |
| En-Suite: | No |
| : : | : : |
| Price | £75,000 |

| 3 LR | *Indices* |
|---|---|
| Location: | SM-1 |
| B-Rooms: | 3 |
| Age: | Modern |
| Rec-Rooms: | 2 |
| Kitchen: | Large |
| Rear-Acc.: | Yes |
| Tot-Area: | >1,200 |
| En-Suite: | Yes |
| : : | : : |
| Price | £98,000 |

Fig. 1. Example cases from Rachmann, the property valuation system

## 3 CBR and Problem Complexity

The basic tenet of CBR is that, rather than solve a problem from first principles, it may be easier to retrieve a similar problem and transform the solution to that problem. In Fig. 2 we attempt to illustrate these trade-offs graphically. SP' represents the specification for a new problem and SL' is the solution to that problem. FP' represents the search process that establishes this solution from first principles—the task we wish to avoid. A CBR solution is worthwhile if the retrieval task R, and the adaptation task A are simpler than FP'.

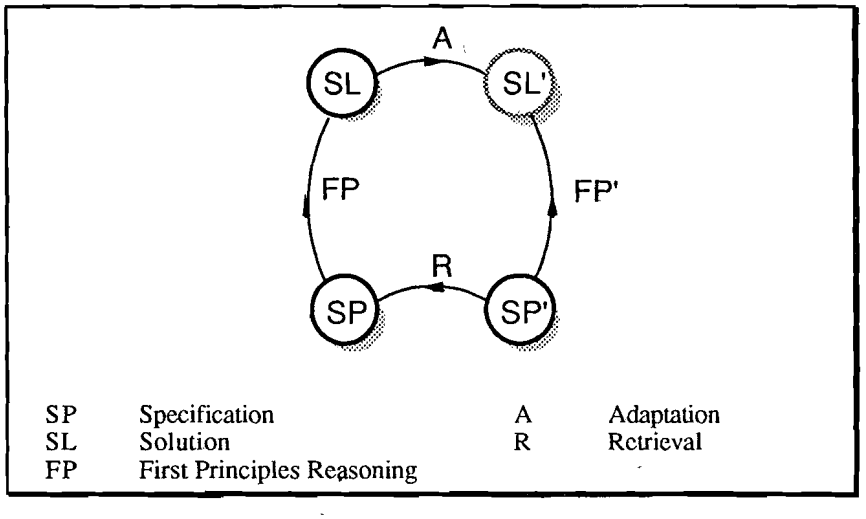



Figure 2. The transformation processes in CBR and in reasoning from first principles.

From an 'AI as Engineering' perspective the big issue here is the complexity of the adaptation task A. An analysis of the CBR literature suggests that CBR adaptation might be divided into three categories arranged in order of increasing complexity as follows:-

- **Substitution Adaptation:** This is the simplest type of adaptation and merely involves substituting some of the parameters in the solution.
- **Transformational Adaptation:** This adaptation is more complex and will involve structural changes to the solution.
- **Generative Adaptation:** This is the most complex adaptation and is not perfectly represented by the diagram. The adaptation process involves a rework of the reasoning process FP in the context of the new problem situation represented by SP'. Derivational Analogy fits in this category.

The Rachmann system shown above is an example of a CBR system using substitution adaptation. This is at the easy end of the spectrum and the advantage of CBR is evident here. For substitution adaptation to work it is necessary that the expression of solutions should be simple. Solutions should be atomic (a single price or the name of a faulty component) or should be made up of a few features with little interdependency. In tasks where the solution expression has a complex structure the adaptation process is more delicate and interaction between solution components must be considered. The cases in CoBRA have this kind of complexity.

Adaptation of complex solutions requires an adequate domain model. So the question is; will this model need to be as complex as that required for a 'conventional' knowledge-based system? Does the use of CBR in complex problem domains manage to avoid any of the knowledge engineering needed for a solution based on planning?

### 3.1 The CoBRA problem domain

CoBRA is an example of a CBR system operating in a domain where substitution adaptation is not adequate. It is a system for creating physical models in engineering analysis. The task being addressed is the generation of simplified models suitable for numerical analysis. This process of model simplification is an important initial stage in thermal analysis in engineering. The objective is to produce a simplified geometric model suitable as a basis for a mathematical model. This simplified model must be a reasonable approximation to the actual physical system. For the human designer this process involves a series of assumptions and justifications that produce the simplified model (see [5] for more details).

CoBRA has been implemented to work on a case base of cooling fins, a typical example of which is shown in Fig. 3. Each case is made up of a representation of the basic model, the simplified model and a reasoning trace of the justifications for the transformations in going from the basic to the simplified model. These transformations and justifications are the key component in the case representation. Because of this CoBRA uses generative adaptation involving a re-run of this reasoning trace—rather than transformation adaptation. This adaptation by regeneration is derivational analogy.

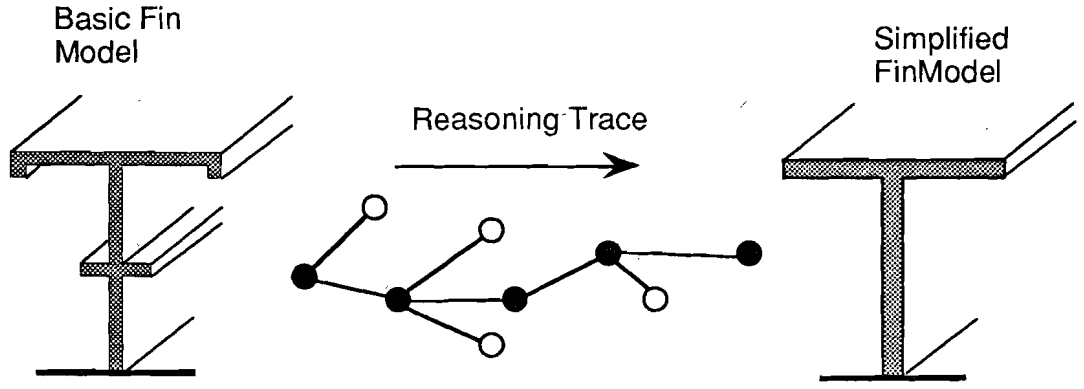



Fig. 3. Example case from CoBRA, the thermal modelling system

### 3.1 Derivational Analogy

Evidently a case in a DA system will have three components; a description of the start state, a description of the goal state and the reasoning trace that produces the goal state. The adaptation process attempts to reinstantiate these reasoning traces in the target case.

In CoBRA each reasoning trace has an action part and a decision part (after [5]). The decision part contains:-

- Alternatives considered and rejected
- Reasons for decisions taken
- Starts of false paths
- Dependencies of later decisions on earlier ones

The action part holds the steps taken as a result of the reasons held in the decision part. A typical action is, "Remove the extended surface which faces into the flow". The two main actions in CoBRA are REMOVE and RESIZE. The actual functions used to express these actions must be sufficiently abstract to allow their application to cases similar to the one with which they are stored. Both the decision and action parts operate on parameters which are common to all cases, for example:

- **altitude:** the altitude of a feature

- **surface-area:** the heat transfer surface area
- **base-area:** the surface area of the feature base.

Summarising, each reasoning trace is made up of a set of actions and justifications for those actions. The reasoning trace can be reinstantiated for the new case if these justifications are valid in the new situation.

## 4 Reconstructive CBR in CoBRA

For Derivational Analogy each case must have three components; a description of the start state, a description of the goal state and the reasoning trace that links these two states. In CoBRA the representation of the start state and the goal state are similar. After all one is a simplification if the other. Figure 4 illustrates a portion of such a case. The diagram on the left shows a cross section of a finned heat exchanger unit and the task addressed by CoBRA is to produce simplified models of cases of this type. The frame description on the right illustrates the representation that is manipulated by the system. A target case contains only this frame representation; this is the problem specification.

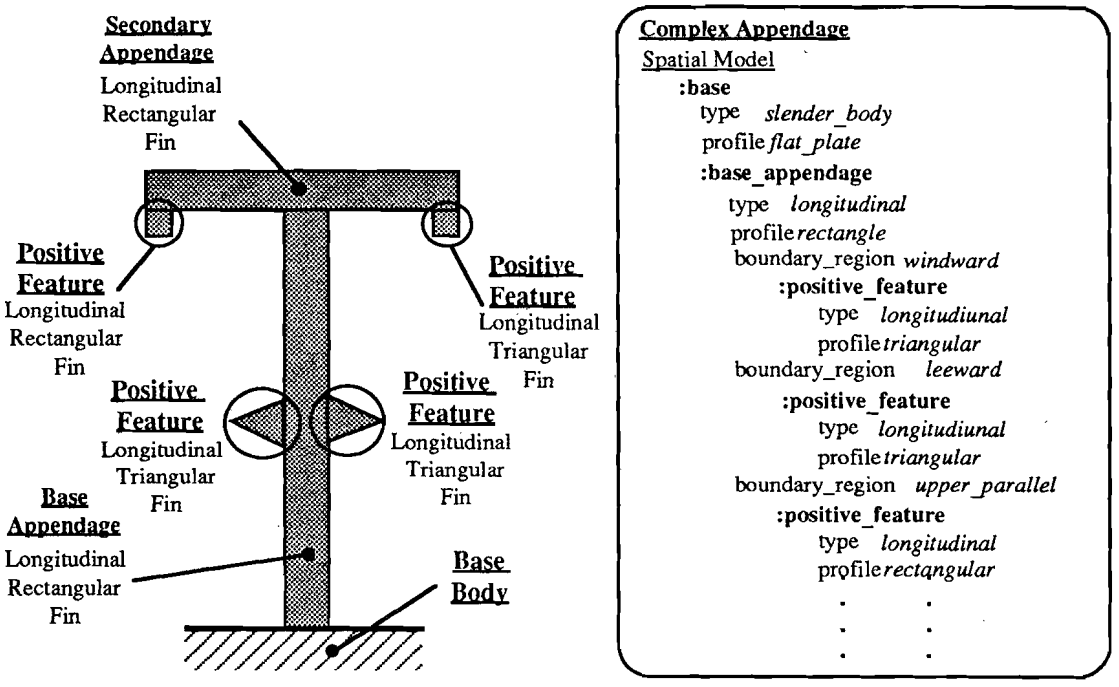



Fig. 4 A spatial classification of the problem with a partial description of the associated target case.

Each base case contains a solution in addition to this problem specification. The solution is made up of the simplified model and the reasoning trace. A typical reasoning trace is shown in Fig. 5. Each node in the reasoning trace represents a decision point in the model simplification process. For instance **Goal_2** considers the heat transfer associated with a sub-feature. There are three possible actions depending on the amount of heat transfer associated with the feature. This reasoning trace encodes the removal of the feature. This illustrates how the reasoning trace in derivational analogy represents a known good route through a vast search space. So whatever about the knowledge acquisition advantages of DA it has a clear advantage in reducing backtracking in problem solving (this point is made by Mostow in [3]).

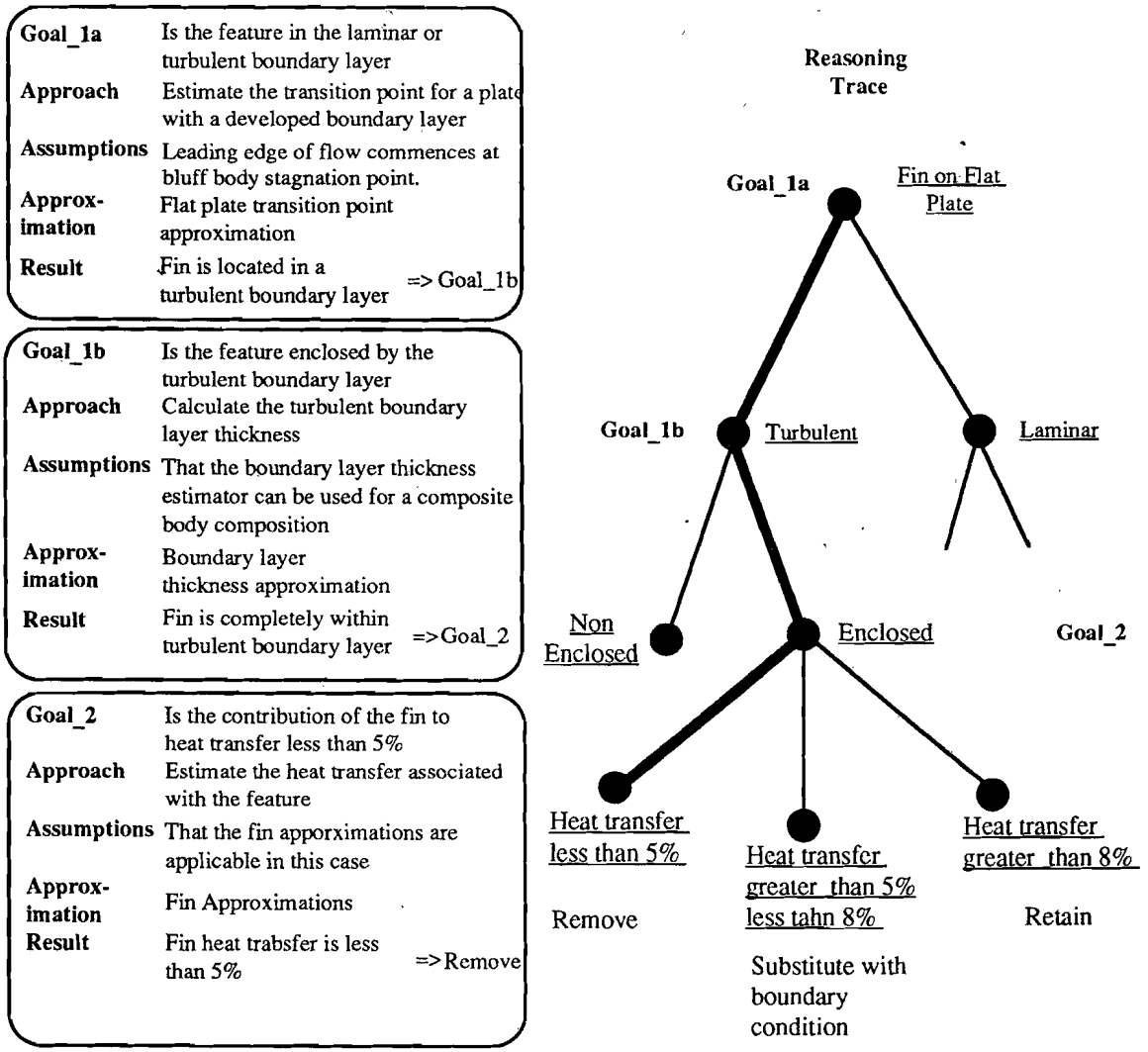



Fig. 5 A derivational trace for a windward finned appendage

## 5 Alternative Knowledge Based Solutions

There has been much AI research on problems of this type that does not use CBR - particularly under the headings of model based reasoning and qualitative reasoning (see for instance [6] [7] [8] This research emphasises representation but it is evident that the reasoning process in mind is one of planning - search with backtracking through a solution space. It should be evident that CBR will help constrain this search process but the question here is whether it will reduce the knowledge acquisition problem? If we consider what a planning based system for our model simplification task would look like we will see that it will not.

The conventional alternative to the CoBRA system involves developing a model of the entities in the problem domain and encoding heuristics that represent the transformations on these entities. The entity model will be the similar to that used is CoBRA and the planning heuristics will be the same as those encoded in the reasoning trace.

The development of a planning system for model simplification requires a knowledge level analysis of the problem domain to ascertain the appropriate transformation heuristics. It is our conclusion from developing CoBRA that the encoding of the reasoning trace is the same kind of task. In this problem domain a comprehensive DA system will explicitly encode the same heuristics as a rule-based planner.

However, it might be said in favour of CBR that the emphasis on cases will focus the knowledge acquisition process.

### 5.1 Alternatives to Rachmann system

The situation is quite different with the property valuation task addressed by the Rachmann system. Setting up the Rachmann case-base requires very little knowledge level analysis. The heuristics are encoded implicitly in the cases. By contrast a rules-based system for the same task would require the determination of the influence features like location, facilities, etc. have on price. So, for this property valuation problem, CBR has avoided this need to explicitly encode a domain model.

## 6 Conclusions

From the perspective of the design and modelling task, CBR does offer some advantages compared to a model based reasoning approach. Establishing cases on the basis of fundamental modelling scenarios allows retrieval of modelling solutions that can be adapted in a focused manner by using derivational reasoning. This avoids extensive backtracking associated with rule based systems. Secondly, it has been our experience that for a complex domain such as convection heat transfer, the process of knowledge acquisition based on reasoning traces provided no special difficulties for our domain expert. This is in marked contrast to our experiences for knowledge elicitation in model based reasoning systems. Finally, because of the episodic nature of the derivational trace, the explanation of the reasoning processes is somewhat more clear.

On the other hand from a CBR perspective, we realise that CBR has been embraced by researchers in knowledge based systems because it has two significant advantages. The first is that case-bases are easier to set up than other knowledge representations. The second is that, in problem solving, cases encode known good routes in the solution space and thus reduce backtracking. These advantages will only be maximised in CBR systems where solution representations are not made up of complex interacting components and the adaptation process is comparatively simple. Out experiments with the Rachmann system confirms this view. However, in CoBRA, where solutions have complex representations, adaptation is more difficult and a full domain model is required to support this adaptation, our opinion is that some of these advantages may have been lost.

Finally we would like to add, that, where automated adaptation is complex and therefore negates some of the strengths of the CBR approach, we believe that involving the user as an agent in the adaptation process as advocated by Kolodner [9] may overcome this disadvantage. We hope to explore this issue in future work.

## References


1. Smyth B., Cunningham P., "Déjà Vu: A Hierarchical Case-Based Reasoning System for Software Design", in *Proceedings of 10th. European Conference on Artificial Intelligence*, Vienna, Austria, ed. Bernd Neumann, Wiley & Son, pp587-589, 1992.

2. Carbonell J.G., "Derivational Analogy: A theory of reconstructive problem solving and expertise acquisition", in *Machine Learning Vol. II*, R.S. Michalski, J.G. Carbonell, T.M. Mitchell eds., pp371-392, 1986.

3. Mostow J., "Design by Derivational Analogy: Issues in the automated replay of design plans", *Artificial Intelligence*, pp119-184, Vol. 40, 1989.

4. Bhansali S., Harandi M.T., "Synthesis of UNIX programs using derivational analogy", *Machine Learning*, pp7-55, Vol. 10, 1993

5. D. Finn, S. Slattery, P. Cunningham, "Modelling of Engineering Thermal Problems - An Implementation using CBR with Derivational Analogy" in *Proceedings of European Workshop on Case-Based Reasoning*, Kaiserslautern, Germany, November, 1993.

6. K. Yip: "Model simplification by asymptotic order of magnitude reasoning" In: D. Weld (ed.): *Working Papers Qualitative Reasoning '93 (QR '93)*. Seattle: University of Washington 1993, pp. 266-272.

7. S. Ling, L. Steinberg, Y. Jaluria: MSG: "A computer system for automated modelling of heat transfer" *Artificial Intelligence in Engineering Design, Analysis and Manufacturing (AI EDAM)*, 7(4), 1993.

8. S. Addanki, R. Cremonini, J. Scott Penberthy: "Reasoning about assumptions in graphs of models". In: D. Weld and J. de Kleer (eds.): *Qualitative Reasoning about Physical Systems. San Mateo: Morgan Kaufmann 1990*, pp. 546-552.

9. Kolodner J.L, "Improving Human Decision Making Through Case-Based Decision Aiding", *AI Magazine*, Vol. 12, No. 2, Summer 1991, pp52-68.

# Modelling of Engineering Thermal Problems - An Implementation using CBR with Derivational Analogy

Donal Finn
Hitachi Dublin Laboratory, O' Reilly Institute, Trinity College, Dublin 2, Ireland.

Seán Slattery and Pádraig Cunningham
Dept of Computer Science, Trinity College, Dublin 2, Ireland.

**Abstract** An interactive cased based reasoning tool for assisting engineers with the mathematical modelling tasks associated with the analysis of thermal problems is described. By representing fundamental thermal modelling scenarios as cases, complex physical systems are modelled in a piecewise fashion by successive application of matching cases. Retrieval is based on the use of qualitative indices, derivational analogy allows for generative adaptation of retrieved cases, thereby providing a basis for validating cases in the context of the problem under consideration. This work represents an alternative perspective to model based reasoning approaches that have been applied to model generation to date.

## 1 Introduction

This paper describes work in progress which aims to develop an interactive case based reasoning system that assists engineers with the mathematical modelling tasks associated with convection heat transfer analysis. This domain is described mathematically by the thermal partial differential equations (PDEs) and is nowadays usually analysed using numerical simulation techniques such as the finite element method. Mathematical modelling precedes numerical analysis and involves abstracting a mathematical model from a real world problem. This is achieved by applying physical and mathematical idealisations, so as to create a model that is computationally realistic to solve, but, at the same time, still retains the important features of the physical system [1,2]. It is for this modelling task that we propose a case-based reasoning solution.

The case base is made up of episodes that represent valid model simplifications. Each case consists of a model that is close to the real world problem, a simplified but valid model of this physical system and a set of assumptions and transformations involved in producing this simplified model. These assumptions and transformations are a key component of the case representation and entail the use of generative adaptation in using retrieved cases. This is the derivational analogy approach to CBR as advocated by Carbonell [3].

The paper is organised as follows; firstly we describe the domain of convection heat transfer by examining the various issues associated with mathematical modelling. Next we discuss from a modelling perspective the conceptual approach that we have taken so that case-based reasoning techniques could be applied effectively. We then discuss implementation work carried out to date and demonstrate an early prototype system called CoBRA (Case-Based Reasoning Assistant) that focuses on spatial modelling. Finally we discuss the use of derivational analogy techniques and describe the structure and contents of a typical reasoning trace.

## 2 Modelling in Heat Transfer Convection

Convection heat transfer problems can be defined as physical systems where heat transfer occurs between a solid body and a surrounding fluid medium, each at a different temperature. Numerical analysis of convection problems is usually carried out in number of stages which have been identified as follows [1]:

**Behavioural Analysis** This is normally the first task in any analysis episode and it involves reasoning about the physical system with the objective of obtaining a behavioural understanding of the underlying phenomena.

**Physical and Mathematical Modelling** This phase involves applying idealisations and simplifications to various spatial and phenomenological aspects of the physical system with objective of abstracting an analysis model. This task is the focus of the current work.

**Numerical Simulation** This phase involves simulating the mathematical model by applying numerical techniques such as the finite element method.

**Visualisation** This stage involves post processing and visualising of the numerical data produced by the simulation process

In this paper, we focus on task of creating an analysis model (physical and mathematical modelling) which is representative of the physical system. We assume that the engineer has already obtained a behavioural understanding of the physical system[1] and consequently, this task is not addressed in this work. The main objective in analysis modelling, is to abstract a mathematical model acting on a domain, that is computationally realistic to solve whilst at the same time preserves the essential integrity of the physical system. We consider construction of an analysis model to have two aspects; a physical perspective and a mathematical perspective [4]. Physical modelling focuses on spatial or geometric aspects of the problem domain and involve applying

[1] Much work to date in qualitative physics has focused on predicting the qualitative behaviour of domains described by ordinarily differential equations (ODEs). However, little work has been carried out on problems defined by partial differential equations.

modelling strategies such as; taking a two dimensional idealisation of a three dimensional physical system, applying geometric symmetries or carrying out feature modelling. Strategies used in feature modelling, illustrated in Figure 1, can involve either replacing an existing complex feature with a simpler feature, removing the feature and substituting it with an equivalent boundary condition or removing the feature completely without any compensatory measures.

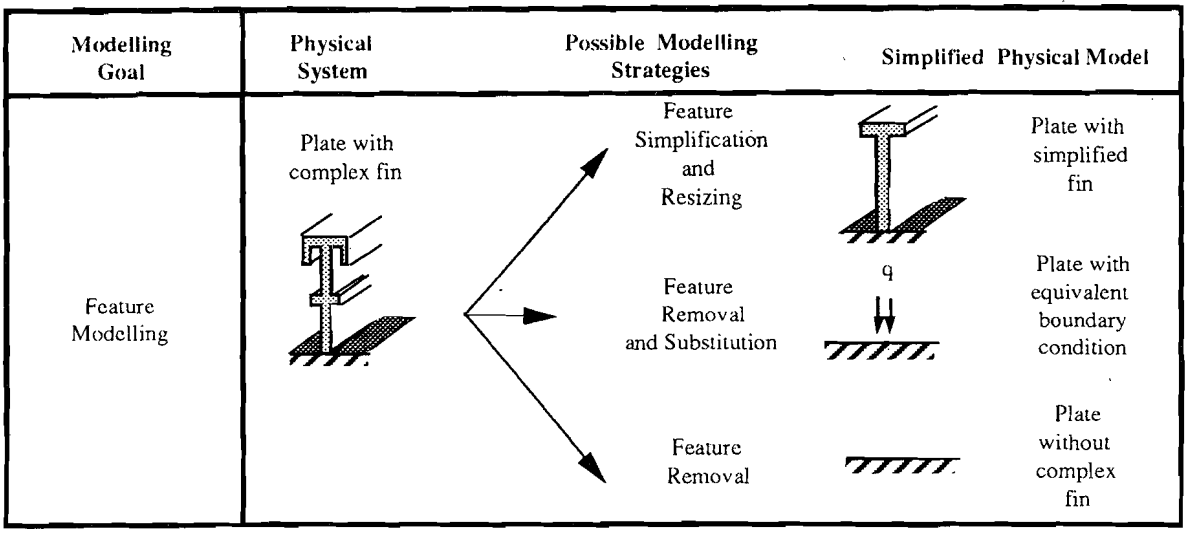



Fig. 1 Feature modelling strategies

Mathematical modelling deals with the construction of a PDE model that describes the thermal heat transfer process. Considering the full thermal PDE, it consists of three sub-equations based on the physical laws of conservation of mass, momentum and energy. Each sub-equation is in turn composed of terms, where each term describes a particular sub-phenomenon. For example, in the energy equation, the diffusion term describes the heat transfer at a molecular level, the advection term describes heat transfer due to bulk motion of the fluid, whereas the viscous dissipation term describes the conversion of mechanical energy to thermal energy due to internal friction effects. In many heat transfer problems it is not necessary to model all these sub-phenomena and therefore terms can be either simplified or even be ignored completely. Another mathematical modelling task (illustrated in Figure 2) is the specification of an analysis volume that defines the extent of the fluid medium to be examined. Although this modelling task has spatial connotations, its specification is essentially governed by type phenomenological analysis that is required by the user.

## 3 Related Work, Conceptual Matters and Design Issues

### 3.1 Related work from heat transfer modelling

To our knowledge, no other work with a similar focus and approach has been undertaken to date. However, three related projects that exploit alternative knowledge based techniques in comparable domains are relevant and are briefly discussed here. Ling and Steinberg [5] describe a system that is currently under development which is aimed at modelling conduction heat transfer problems. Model based reasoning is the basis for the approach taken in this work. The system is implemented as part of a greater design system and emphases is placed on achieving automated modelling decisions without the intervention of the user. Three modelling issues are dealt with and these include; the choice of control region, the determination of the relevant physical processes and the abstraction of appropriate mathematical equations. However, from a geometric perspective, the coverage of this system is confined to simple parallelepiped domains. In addition its confinement to conduction based problems makes this domain considerably simpler than convection heat transfer problems. Wentorf and Shephard [2] describe a rule based expert system, that deals with idealisation issues associated with modelling in stress analysis of aircraft. The emphasis in this system is the use of knowledge based techniques to integrate and control interdisciplinary tools in an analysis system such as CAD interfaces, error optimisers and numerical error predictors. Finally, Yip describes a system for simplifying the Navier Stokes (fluid flow) equations using order of magnitude reasoning within a qualitative analysis framework [6]. This system produces idealised PDE models which are mathematically complete, but in many cases have no physical meaning and may sometimes be computationally insolvable. Nevertheless, this work is important as it examines how PDE type problems can be tackled using qualitative physics.

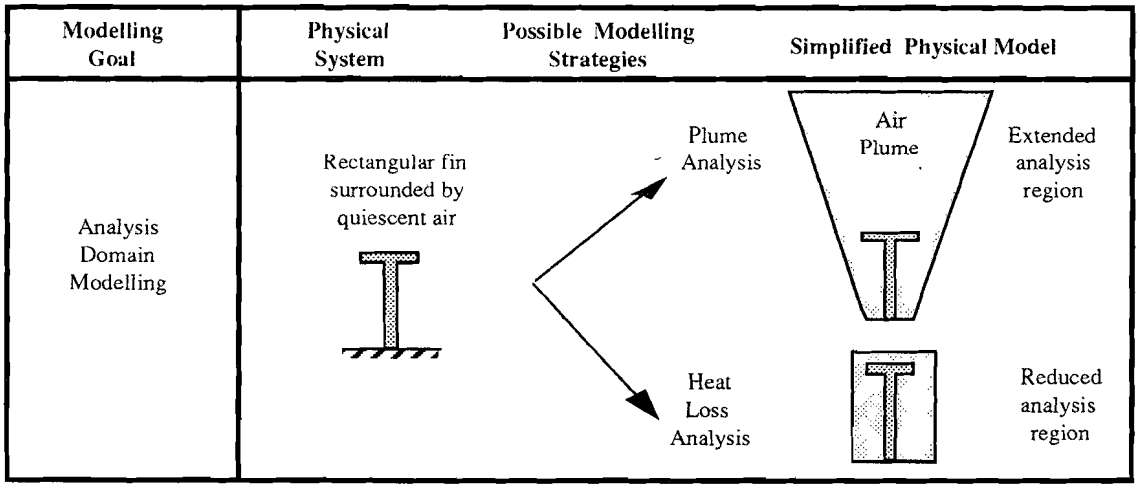



Fig. 2 A subset of mathematical modelling issues

### 3.2 Conceptual Issues

In this work, particular attention was given to observing how engineers model convection heat transfer problems. These observations have strongly influenced the approach adopted and are summarised here;

- Engineers usually model complex convection problems in distinct stages. These stages correspond to the physical and mathematical modelling issues outlined in Section 2 and are as follows; spatial modelling, phenomenological modelling, dimensional reduction, temporal modelling and control volume modelling.
- Engineers exploit a number of techniques when modelling convection problems, these include; the use of first principle domain knowledge to reason about modelling strategies, exploitation of previously modelled problems and relying on the guidance from more experienced colleagues. In most modelling episodes, a combination of these techniques are used.
- When investigating a particular modelling stage, e.g., spatial modelling, engineers usually decompose a complex physical system into easily understood sub-problems. These sub-problems are sufficiently low-level to be related to what we call classical engineering modelling scenarios. A scenario typically consists of simple modelling episodes and allow engineering approximations and heuristics to be applied, thereby permitting the modelling issue under consideration to be evaluated easily.

These conclusions influence our approach in two ways; firstly, for an interactive system it is imperative that we aim to accommodate the end-user and therefore the system should attempt to integrate with the modelling patterns used by engineers. Secondly, by capturing engineering first principles, engineering approximations and heuristics within fundamental classical modelling scenarios, it is possible to build a case based reasoning system that is based on episodic based templates that provide guidance for modelling tasks.

### 3.3 Design Approach adopted in this work

We summarise here our conceptual approach to modelling which forms the basis for the implemented CBR system.

- The system is organised so as to allow modelling to be carried out in distinct stages. In this paper, we consider the stage of spatial feature modelling.
- Within any modelling stage, modelling decisions are taken in a piece wise fashion by examining each modelling issue in turn.
- Case based reasoning with derivational analogy techniques form the core approach. Cases are based on fundamental modelling scenarios and are derived from episodic modelling events.
- Solutions within cases describe a model strategy that can be applied to similar target cases. The strategy is usually in form of some action which is in response to a particular modelling goal.
- Derivational traces describe the full engineering reasoning basis by which a particular modelling solution was reached. They also act as an explanation facility and validator of the case solution. More importantly however, they allow solutions of base cases that are close to the target case to be adapted and applied to the target.

## 4. Implementation Details

### 4.1 A Convection Heat Transfer Problem

Figure 3 illustrates a typical convection heat transfer problem that can be tackled by the modelling system. The physical system consists of a finned heat exchanger tube that dissipates heat to the surrounding ambient air. Two

complex appendages are attached to the cylindrical base, each appendage has additional minor associated features. The modelling goal in this task is to assess the importance of both the minor features and the appendages themselves, this task corresponds to the spatial modelling phase described in Section 2.

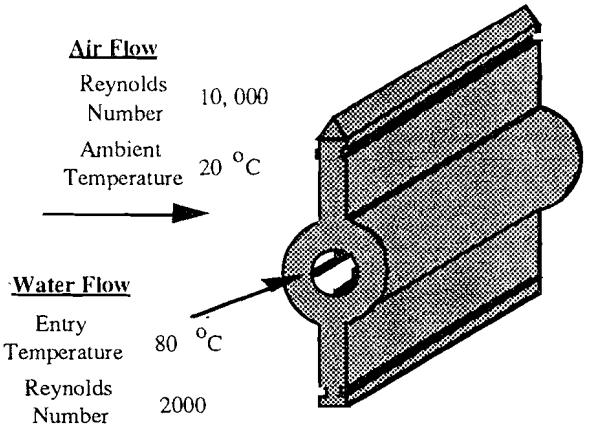



Fig. 3 A finned heat exchanger tube

### 4.2 Target Case Description

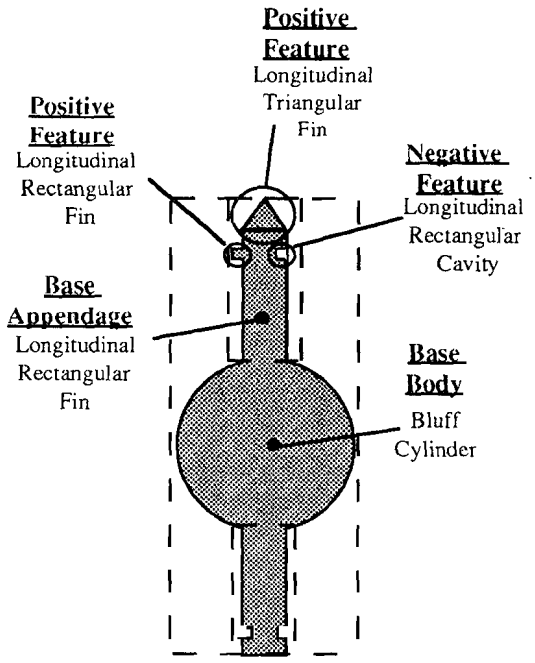
A target case consists of a frame based representation of the physical system. Within a target frame, representation is organised according to the different modelling perspectives; spatial modelling, phenomenological modelling and control volume analysis modelling. Figure 4 illustrates from a spatial perspective how the finned heat exchanger is classified and shows some of the indices used to describe the problem. In this case, a partonomic type relationship at three levels describes the essential components of the physical system, namely; the base cylinder, the complex appendages and their associated minor features. In this problem, the base is classified as a cylindrical bluff body in crossflow, the complex appendage is a rectangular longitudinal fin with features located on its windward, upper parallel and leeward sides. These features are a longitudinal rectangular cavity, a longitudinal triangular fin and a longitudinal rectangular fin. Problem parameters such as geometric data are also included in the target case but are not used as indices, however this information is used in the derivational traces.

### 4.3 Modelling Approach and Base Case Description

In Section 2, we argued that engineers normally model convection problems by decomposing the problem into well understood scenarios and considering each of these in a sequential manner. By classifying the heat exchanger problem as shown in Figure 4, this decomposition has been effectively achieved. Modelling progresses by firstly examining the role of the minor features with respect to the complex appendage and secondly the role of the complex appendage with respect to the cylindrical base. Each of these modelling episodes are sufficiently fundamental, so that they are comparable in terms of complexity and detail to the classical modelling scenarios discussed in Section 3. Consequently, all base cases are represented at this modelling abstraction level. Figure 5 illustrates one base case, that of modelling a longitudinal positive rectangular feature on the windward side of a rectangular fin. This base case is a classical heat transfer situation, is well understood and can be adapted and applied to a range of similar problems. In this base case, qualitative indices describe the minor feature and the associated base appendage. The modelling action or solution associated with this base case is that the feature can be removed completely without the need for any compensatory action. However, this action is not applied directly but is instead implemented by a process of regenerative transformation by applying the associated derivational trace.

### 4.4 Matching and Mapping

Case retrieval is implemented in a two stage process, matching (or base filtering) and mapping. In our initial prototype matching is implemented using an activation net which is made up of activation units which correspond to the indices of base cases. A feature vector is created for each target case which contains the relevant indices of the problem. The feature vector is the basis by which the activation units are initialised and on completion, each case in the case base is contains a value of how many indices it shares with the target case. The mapping stage is concerned on establishing the correspondences between the base cases and the target cases. In our initial prototype, mapping based on establishing the full set of matching features between the target and base cases is the criteria for retrieving useful cases.




**Bluff Cylinder with Complex Appendages**

Spatial Model

**:base**
type *bluff_body*
profile *cylindrical*
**:c_appendage**
type *longitudinal*
profile *rectangle*
boundary_region *windward*
**:positive_feature**
type *longitudiunal*
profile *rectangular*
boundary_region *upper_parallel*
**:positive_feature**
type *lomgitudinal*
profile *triangular*
boundary_region *leeward*
**:negative_feature**
type *longitudiunal*
profile *rectangular*
boundary_region *lower_parallel*

Fig. 4 A spatial classification of the problem with a partial description of the associated target case.

### 4.5 Derivational Traces

Derivational traces are exploited in this domain, because, although the target and base cases may map qualitatively, small differences between physical parameters such as spatial or medium data can lead to significantly different solutions. Such differences cannot be expected to be captured in the initial qualitative classification of the problem, furthermore, to index all episodes based on both descriptive and parametric indices would result in an intractably large case base. A derivational trace describes the basis of the modelling solution, in this example, the removal of a windward longitudinal feature on a rectangular appendage, the reasoning behind these decisions and the engineering approximations and heuristics used in the evaluation process. In this example, the solution in the base case was derived in two ordered stages; firstly, the influence of the feature on the medium flow field was determined and found to be negligible and secondly the contribution of the feature to total appendage heat transfer was assessed and found to be of minor importance. Figure 6 shows a simplified version of the derivational trace. The fist stage examines the influence of the feature on the flow field and consists of Goals 1a and 1b. This involves determining whether the feature is actually fully contained within a turbulent boundary layer, and if so, the influence of the feature on the flow field is deemed negligible. Goal examines the contribution of the feature to overall heat transfer. In the base case, the heat transfer contribution of the feature was of the order of 4% of total heat transfer well within the 5% constraint, so therefore the fin was be removed, in the target case, this contribution was of the order of 3.5% thereby permitting the feature to be removed.

**Rectangular Appendage with Fin**

**Description**
direction: *windward*
type *flat plate*
**:positive_feature**
type *longitudinal*
profile *rectangular*
location *aft*
flow_regime *turbulent*

**Solution**
Action *Remove fin*
Compensation *None*

**Derivational Trace**

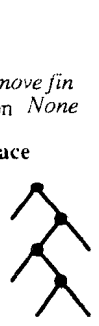


Fig 5 A sample base case for modelling a rectangular appendage with a positive feature

## 5. Conclusions

We have described a preliminary prototype of an interactive case based reasoning tool for mathematical modelling of thermal engineering problems. Derivational analogy techniques are exploited to provide for generative adaptation and validation of base cases. We have found that because of the complexity of the domain, derivational analogy techniques are necessary to provide for case adaptation and validation. Nevertheless we believe that this work represents an important alternative perspective to model based reasoning approaches that have been applied to model generation to date.

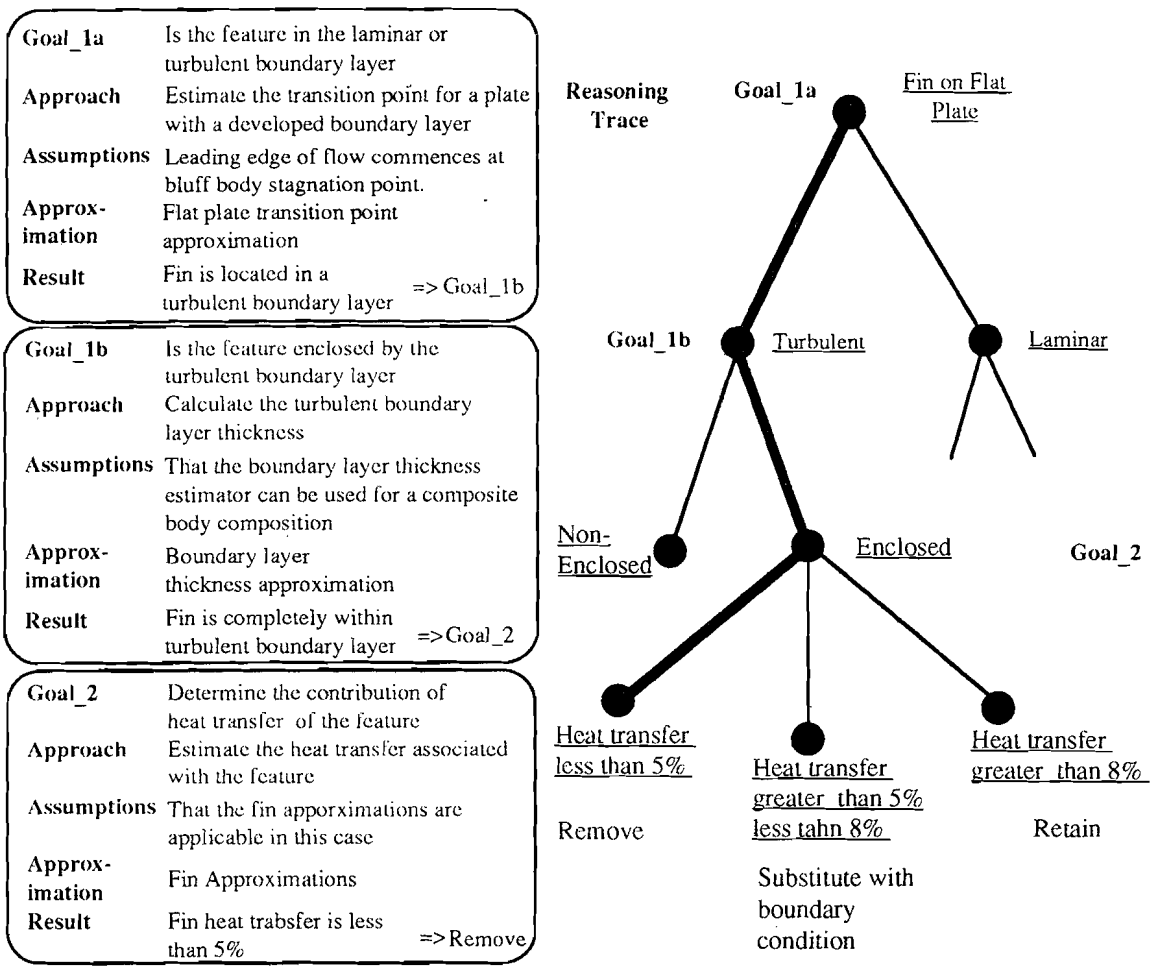



Fig 6 A derivational trace for a windward finned appendage

## References


1. K. Bathe, N. Lee, M. Bucalem: On the use of hierarchical models in engineering analysis. Computer Methods in Applied Mechanics and Engineering, 82(1-3), pp. 5-26, 1990.
2. M. Shephard, P. Baehmann, M. Georges, E. Korngold: Framework for reliable generation for the control of analysis idealisations. Computer Methods in Applied Mechanics and Engineering, 82 (1-3), pp. 257-280, 1990.
3. Carbonell J.G: Derivational Analogy: A theory of reconstructiuve problem solving and expertise acquisition. In: R.S. Michalski, J.G, Carbonell, T.M. Mitchell (eds)., Machine Learning, 2 pp , 371-392, 1986.
4. D. Finn, J.B. Grimson, N. Harty: An intelligent modelling assistant for preliminary analysis in design: In: J. Gero (ed) Proc. of the 2nd International Conference on Artificial Intelligence in Design, pp 579 - 596, Carnegie Mellon University, Pittsburgh, June 1992.
5. S. Ling, L. Steinberg. Approximation operators in distributed computing: In: D. Weld (ed): Working Papers Qualitative Reasoning '93 (QR '93). Seattle: University of Washington 1993, pp. 138-144.
6. K. Yip: Model simplification by asymptotic order of magnitude reasoning. In: D. Weld (ed): Working Papers Qualitative Reasoning '93 (QR '93). Seattle: University of Washington 1993, pp. 266-272.

# Reformulation in Analogical Reasoning

Erica Melis
Universität Saarbrücken, FB Informatik
66041 Saarbrücken, Germany
email: melis@cs.uni-sb.de

## 1 Introduction

In many domains there are different representations of the same cases. For instance, the filler for the time slot in a case description can be made in terms of hours or in terms of morning, noon, afternoon, evening, and night; a kinematic description can be made in terms of mass and velocity, as well as in terms of momentum and energy. Actually some approaches and techniques in theoretical computer science and artificial intelligence are (implicitly) dealt with reformulation. Reformulation was also identified in [4] as one of the more difficult issues in analogical reasoning.

However, the use and techniques of reformulating the base and the target problem for analogy formation have found little attention although Indurkhya [5] and Russell [7] established the importance of of reformulation in analogical reasoning in general. In [6] we have shown that reformulation rather than only symbol mapping is often necessary for advanced analogy-driven theorem proving in mathematical domains. There we have developed techniques for reformulation within a proof planning context. As we think that the results can be generalized for analogical reasoning and case-based reasoning, we give an outline here that is, of course, restricted by the lenght of the paper.

## 2 The General Problem of Reformulation

- The reformulation of a case representation can serve to identify *explicitly* the similarities of cases with *only implicitly shared aspects*. Since case-based and analogical reasoning are based on the similarities of problems the machine supported reformulation of a given representation is often paramount for the solution of the problem.
- We consider reformulation as a change of the representation of problems *and* solutions (goals). This is possible because of the connections between goals and goal-relevant aspects of the problems.
- Reformulation is different from the modification/adaptation step of analogical reasoning:
  - It is done during retrieval or, if the analogous case is given, before matching the problems.
  - Some reformulations affect the base *and* the target case.
  - Modifications are partially anticipated by the reformulation.
- Mechanisms of reformulation have a domain-dependent search space and domain-dependent control strategies guiding the choice of reformulations.

## 3 Reformulation in Analogy-Driven Theorem Proving

We consider a situation, where a proof $\mathcal{S}$ of a base problem $\mathsf{S} = (ass_{\mathcal{S}} \vdash thm_{\mathcal{S}})$ is given, and the task is to find a proof $\mathcal{T}$ of the target problem $\mathsf{T} = (ass_{\mathcal{T}} \vdash thm_{\mathcal{T}})$ which is supposed to be analogous

to $\mathcal{S}$.

Our approach is embedded in a proof planning framework (see [2]). It considers plan operators, called methods, as basic units. Methods encode problems as well as partial proof schemata. They are represented by frame-like structures, as for example,

| Method: **hom1** | | |
|---|---|---|
| parameter | **formula**$_f$, **f**: function | |
| pre | {ass(1), ass(2), ass(3)} | |
| post | ; $symmetric(\mathbf{f}(\rho))$ | |
| dec-cont | 1. ; 1 ⊢ $\forall x$ **formula**$_f$ (LEMMA)<br>2. ; 2 ⊢ $\forall\sigma(symmetric(\sigma) \rightarrow \forall x,y((x,y) \in \sigma \rightarrow (y,x) \in \sigma))$ (LEMMA)<br>3. ; 3 ⊢ $symmetric(\rho)$ (LEMMA)<br>4. ; 1, 3 ⊢ $\forall x,y((x,y) \in \mathbf{f}(\rho) \rightarrow (y,x) \in \mathbf{f}(\rho))$ ($PLAN_1$)<br>5. ; 1, 2, 3 ⊢ $symmetric(\mathbf{f}(\rho))$ ($METHOD_5$ 2 4) | |
| procedure | schema-interpreter | |
| history | | |

The idea of this method is to prove $symmetric(\mathbf{f}(c))$ from certain preconditions and from the definition of **f** in line 1.

All slots but **procedure** have declarative slot fillers. Thus methods can be reformulated by so-called meta-methods that change the declaratively filled slots. We give an example for such a meta-method, called **Add Argument**:

| Metamethod: **Add Argument** | |
|---|---|
| parameter | P: problem |
| pre | P=$(ass;thm)$ and term $\mathbf{f}(x)$ occurs in concl(M) and $thm$=concl(M)$[\mathbf{f}(x)/\mathbf{f}'(x.y)]$ |
| post | M'=M$[\mathbf{f}(t_1)/\mathbf{f}'(t_1,t_2)]$, where $t_1, t_2$, are terms |
| procedure | PROCADD |
| rating | ADD-rating |

**Add Argument** is applicable if the unary function symbol $f$ occurs in the conclusion of M. This meta-method changes a unary function to a binary one. This is coded in PROCADD. **Add Argument** should be applied if concl(M) of the $M_{1*}$-method[1] M equals the conclusion of an $M_{2*}$-method after replacing the unary function symbol $f$ by a binary function symbol $f'$.

## 3.1 The Model

Starting with the method $M_1$ made up from the base problem P1 and its proof, and method $M_2$ made up from the target problem P2 without a proof, the goal is to reformulate $M_1$ to a method $M_{1k}$ in $k$ steps, such that the postcondition of $M_{1k}$ matches P2. Although the reformulation could in principle be limited to P1-methods, such that P1=$(ass1;thm1)$ is reformulated to a problem $(ass1';thm1')$ with $thm1' = thm2$ and $ass1' \subseteq ass2$, it is more convenient to apply normalizing and abstracting meta-methods to both $M_{1*}$- *and* $M_{2*}$-methods. Such reformulations are advantageous since they are more purpose directed: It is easier to abstract two methods and then to find an additional reformulation that yields a problem that matches the abstracted problem, than to find an abstraction, a reformulation, and a reverse abstraction that provide a problem matching the original P2. In the former case it is easy to find out which reverse abstraction to use. Also the reformulation of $M_{1j}$ to $M_{1m}$ is more goal directed (see figure 1).

[1] the $M_{1*}$- and $M_{2*}$- methods are descendent of the base and the target methods, respectively

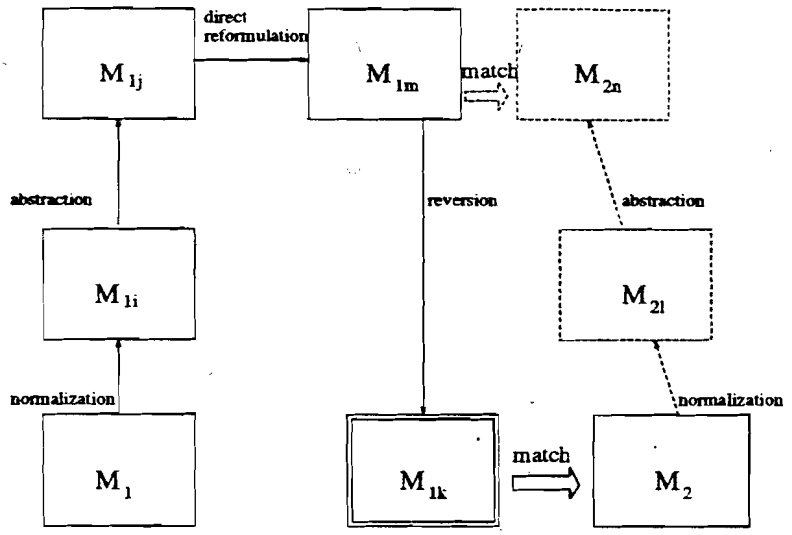



Figure 1: Outline goal directed reformulation

In addition to this process there is a preparation procedure for the verification of methods, which removes the method variables from $M_{1k}$ and yields the method $M_{1r}$. This method $M_{1r}$ is then checked by a verifier and if the verification succeeds, $M_2$ is replaced by $M_{1r}$, which finally contains a verified proof schema.

If the verification fails, the same process is tried again but with all the sub-methods, sub-submethods etc. of $M_1$ and $M_2$. These sub-methods are obtained from the structuring reformulations presented below.

When all methods and sub-methods have been dealt with, there are methods that were obtained from $M_1$ by some reformulations, such that their postconditions match P2 or some subproblems of P2. These successfully reformulated methods serve as preferred candidate elements for the proof plan. Verified methods are favoured candidates, compared to methods that have the same postcondition but are not verified.

## 3.2 Analysis of Reformulation in Theorem Proving

Here we mention some basic properties of our approach. They are due to the domain of mathematical theorem proving.

- The reformulation steps depend on the source problem S and on the target problem T, and to a certain extend on the proof/proof idea for S.
- *Nature of the representational differences between the analogues*
  Problems/proofs can be formulated at several levels of abstraction. Two problems/proofs can be instances of the same abstraction, e.g., proof by Diagonalization Method. Different representations can be due to rewriting w.r.t. (equations of) a theory or just to logical reformulation. Representations can differ in their basic concepts (their signature). There are symmetries and dualities (both are interpretations of a theory in a theory) in mathematics bridging differences.
- *Metamethods available to the system*
  We identified several classes of reformulations which differ in their application and effects:
  NORMALIZATION, with, e.g., `Expand Definitions`
  ABSTRACTION, with, e.g., `Homomorphy Abstraction`

DIRECT REFORMULATION, with, e.g., **Symbol Mapping** and **Add Argument**
STRUCTURING, with, e.g., **Conjunctive Decomposition**
REVERSION, with, e.g., **Reverse Hom-Abstr**.
Our current set of (heuristically) justified metamethods is not complete. Metamethods are to discover by experience in mathematics.

- *Control strategy guiding the choice among the metamethods*
  At any point in time during the reformulation process there may be several meta-methods applicable to more than one method, hence the need for control strategies. A first and important control strategy fixes the right choice of the class of reformulations and these classes are to be activated in a fixed sequence; afterwards we have to pick the heuristically best choice within each class.

  The general sequence of these classes that turned out to be most useful is:

  1. Normalization
  2. Abstraction
  3. Direct Reformulation
  4. Restructuring

  In addition, metamethods have preconditions for their application to be tested. If several metamethods are applicable in the same situation, then their ratings are decisive.

- *How the reformulation leads to a proof plan for $T$.*
  Proof planning tries to partially order the successfully reformulated methods by comparing instances of their pre- and postconditions respectively. It can use information from the structuring of the $M_{1*}$- and $M_{2*}$-methods. Proof Planning starts with a method M that has the desired problem P2 as its postcondition. Then it looks for methods that have problems of pre(M) (maybe less instantiated) as its postcondition etc. The process stops when the preconditions of the new methods are empty or there are no new methods. It may provide several proof plans.

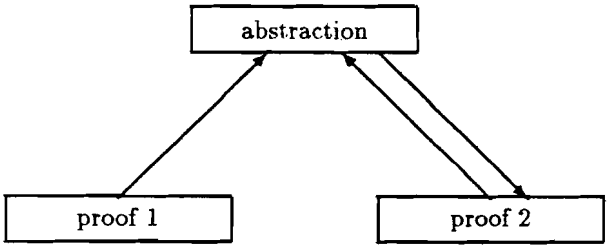
  Often there will still be gaps between the elements of the proof plan. That is, not all preconditions of a method are found in the succeeding methods. Hence, to obtain a plan as complete as possible, additional methods have to be inserted which can be found, for instance, by searching bridge lemmas or by difference matching, see [3, 1].

# 4 Conclusion

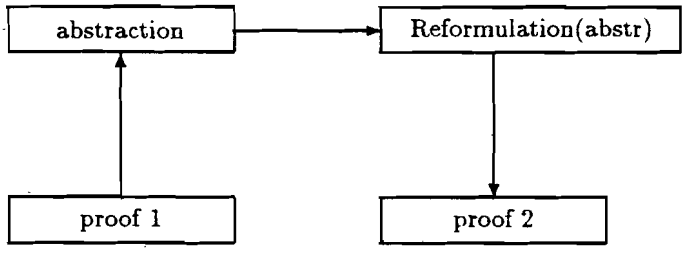
Within this framework several types of analogies can be established:

- Analogies based on direct mapping of proofs onto proofs, as in previous approaches to theorem proving by analogy:

- Analogies based on abstractions of proofs (and subsequent reverse_abstraction):




- Analogies based on abstracted and in addition reformulated proofs with subsequent reverse_abstraction.




# References

[1] D. Basin and T. Walsh. Difference matching. In D. Kapur, editor, *Proceedings 11th International Conference on Automated Deduction (CADE)*, volume 607 of *Lecture Notes of Computer Science*, pages 295–309. Springer, 1992.

[2] A. Bundy. The use of explicit plans to to guide inductive proofs. In E. Lusk and R. Overbeek, editors, *Proc. 9th International Conference on Automated Deduction (CADE)*, volume 310 of *Lecture Notes in Computer Science*, pages 111–120, Argonne, 1988. Springer.

[3] J. Cleve and D. Hutter. A new methodology for equational reasoning. Technical report, Universität des Saarlandes, 1993.

[4] R. Greiner. *Learning by Understanding Analogies.* PhD thesis, Stanford University, 1985. Technical Report STAN-CS-85-1071.

[5] B. Indurkhya. *Metaphor and Cognition.* Kluwer Academic Publisher, Dordrecht, 1992.

[6] E. Melis. Change of representation in theorem proving by analogy. SEKI-Report SR-93-07, SFB, University Saarbrücken, 1993.

[7] S.J. Russell. *The Use of Knowledge in Analogy and Induction.* Pitman, London, 1989.

# Similarity-based Adaptation and its Application to the Case-based Redesign of Local Area Networks

Frank Zeyer and Michael Weiss
{frank,weiss}@pinfo100.informatik.uni-mannheim.de

Lehrstuhl für Praktische Informatik I, Universität Mannheim
D-68131 Mannheim, Germany

**Abstract.** The notion of similarity is important for both indexation and adaptation. But traditionally, research has been almost exclusively focused on the use of similarity between cases for indexation. Therefore, the present paper looks at similarity-based adaptation instead. Two uses of similarity in adaptation are identified: the adaptation of actions prescribed by a case solution using a hierarchy of similar actions and a heuristic weighting of actions where multiple substitute actions have been found for an action in the solution. The approach is illustrated with an example from the domain of local area network redesign.

# 1 Introduction

The case-based problem solving process can be decomposed into two distinct phases: indexation and adaptation [2]. In the first, a case is selected from a case base according to its similarity to the current problem situation. In the second, the solution proposed by the selected case is adapted by applying appropriate changes that reflect the difference between the current problem situation and the assumptions made in the situation on which the case is based.

The notion of similarity is important for both indexation and adaptation. But traditionally, research has been almost exclusively focused on the use of similarity between cases for indexation. This becomes particularily obvious with regard to the list of papers submitted to this workshop. Therefore, the present paper looks at *similarity-based adaptation* instead.

Two uses of similarity in adaptation will be identified: the adaptation of actions prescribed by a case solution and a heuristic weighting of actions where multiple substitute actions have been found for an action in the solution. We introduce the concept of *action hierarchies* of similar actions, where their degree of similarity is measured by both abstraction and specificity.

The paper will proceed as follows. First a typical problem situation will be described. Next the representation of cases will be presented. In the main part of the paper the adaptation mechanism itself will be outlined. It comprises three phases: search for a substitute action, parameter adaptation and application of the identified actions. In conclusion, the present state of the work will be reviewed. Throughout the paper the approach is illustrated with a concrete example from the domain of local area network design.

# 2 Problem Situations

The application domain of the proposed case-based system is the redesign of local area networks (LANs). Such networks are composed of data segments linked by repeaters or

bridges. Each data segment connects to a number of workstations and servers to which peripheral devices are usually attached. Each server provides a list of services that can be requested by the workstations.

The redesign process is initiated by the identification of a bottleneck situation (redesign problem). It is assumed that this redesign problem is obtained from the application of an appropriate diagnosis algorithm. Together, the redesign problem and a description of the local area network provide the input to the system, which is captured in a problem situation.

**Definition 1.** A *problem situation* contains a description of the network and a problem description.

The full formalisim for the description of a local area network will not be presented here. Instead, we will refer to the topology shown in fig. 1.

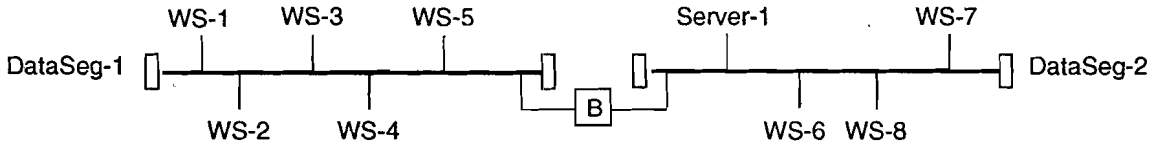



**Fig. 1.** LAN topology for a typical problem situation

**Definition 2.** A *problem description* contains sender, receiver, a description of the kind of traffic between sender and receiver and the path by which they communicate. It also states a performance requirement that must be met by the solution.

A problem description states that for some reason the communication between the sender and the receiver is insufficiently supported by the current network configuration. For example, the problem description below describes a contention of the bridge from the topology in fig. 1:

Sender: DataSeg-1
Receiver: Server-1
Traffic: S(database, Server-1) ⇒ DataSeg-1
D(DataSeg-1) ⇒ Server-1
P(B) = NOT-ACCEPTABLE
Path: DataSeg-1/B/DataSeg-2/Server-1
Required: P(B) = WITH-MARGIN

This problem description is to be read in the following way: The database service of Server-1 is heavily used by clients in data segment DataSeg-1 (the notation "⇒" is used). Large amount of data are flowing from this data segment to the server. The descriptors S and D are used to denote a service offered by a particular server and the data traffic caused by a network component. The other descriptor P symbolizes performance.

# 3 Case Representation

The problem description from the problem situation is now used as in index into the case base. (We will not discuss indexation here, however.)

**Definition 3.** A *case* connects a problem description to a solution description.

A case also prescribes a list of actions, such as duplicate-server, to be performed on the local area network. The list of actions comprises the solution description.

**Definition 4.** An *action* is an operation that changes either the structure or the composition of the network. In addition, actions can be specified with variable components or parameters.

**Definition 5.** A *solution description* is a list of actions that applied resolve the redesign problem associated with the case.

For example, the action duplicate-server creates an additional server in another data segment with the same range of services associated with the original server. The services are represented as an attribute to the server object.

duplicate-server(Server, Segment)

The following is a case that fits the problem situation above. It is composed from the problem description:

Sender: Segment-1
Receiver: Server-2
Traffic: S(print-service, Server-2) $\rightarrow$ Segment-1
D(Segment-1) $\Rightarrow$ Server-2
P(B) = NOT-ACCEPTABLE
Path: Segment-1/R/Segment-2/B/Segment-3/Server-2
Required: P(B) = WITH-MARGIN

and the solution description:

duplicate-server(Server-2, Segment-1)

In this case an average number of print jobs is submitted to Server-2 by clients in Segment-1 (using the notation "$\rightarrow$"). However, since these print jobs cause a high traffic there is heavy load on the bridge B between Segment-2 and Segment-3. The performance of B is therefore rated NOT-ACCEPTABLE. The required performance of B is to operate WITH-MARGIN. In order to achieve this desired state, i.e. to keep the traffic caused by the print jobs local, Server-2 should be duplicated.

# 4 Adaptation Mechanism

The assumptions made in the case problem situation usually do not match exactly those of the current situation. Therefore a substitute action has to be determined that fits these different assumptions. At this point of the process the adaptation mechanism enters the game. The adaptation mechanism comprises three elements:

- search for substitute actions,
- parameter adaptation, and
- application of the selected actions.

These will be described in more detail now. Fig. 2 shows how these elements are related. We now introduce the concept of action hierarchies and our notion of similarity.

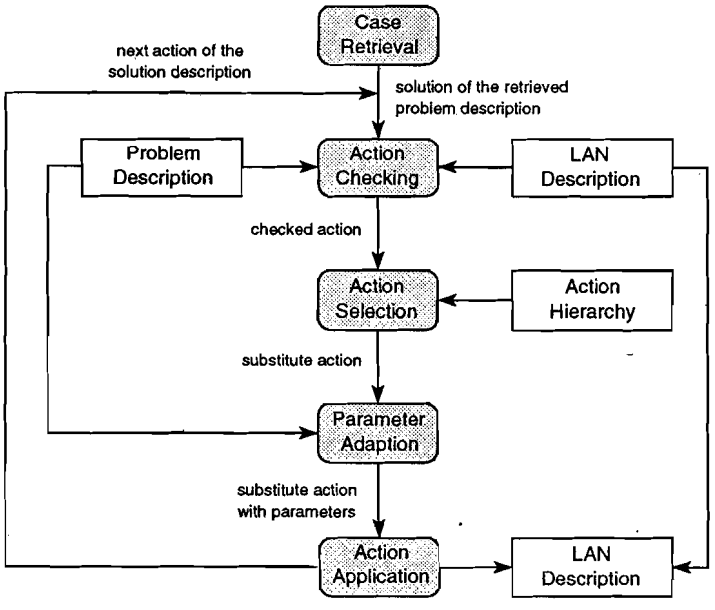



**Fig. 2.** Adaptation cycle

**Definition 6.** An *action hierarchy* is a hierarchy of similar actions, where their degree of similarity is measured by both abstraction and specificity.

Fig. 3 shows parts of action hierarchy. In particular, the left half contains the specializations of the action reposition-server.

**Definition 7.** A *substitute action* is an action that as similar as possible to the action in the solution description of the case, and at the same time suitable.

**Definition 8.** The *degree of similarity* $\mu(p \sim p')$ of an action $p$ and a particular substitute action $p'$ is defined by a weighted difference of the degree of specificity $\mu_+(p \sim p')$ and abstraction $\mu_-(p \sim p')$ of $p'$:

$$\mu(p \sim p') = \mu_+(p \sim p') - \beta\ \mu_-(p \sim p')$$

The parameter $\beta$ (for bias) can be definied to suit the user's pereferences. If the user chooses to favor substitute actions that are closer within the hierarchy to the orginal action, the value of $\beta$ should be high. The reason for this will be intuitive from the example below. The following definitions apply.

**Definition 9.** Let $p$ be an action, $p'$ a particular substitute action and $p \wedge p'$ the least common abstraction of $p$ and $p'$. The *degree of specificity* $\mu_+(p \sim p')$ is the distance $d$ between $p \wedge p'$ and $p'$ in the abstraction hierarchy:

$$\mu_+(p \sim p') = d(p \wedge p', p')$$

The *degree of abstraction* $\mu_-(p \sim p')$ is the distance $d$ between $p$ and $p \wedge p'$ in the abstraction hierarchy:

$$\mu_-(p \sim p') = d(p, p \wedge p')$$

For example, the action duplicate-server could be relaxed either to the action split-server or the action increase-bridge-buffer (compare fig. 3). The latter action is again a specialization of the action improve-hardware. On the one hand it is very much different from the original action, on the other it could be used as substitute action due to its specificity; the choice depends on the value of the bias $\beta$.

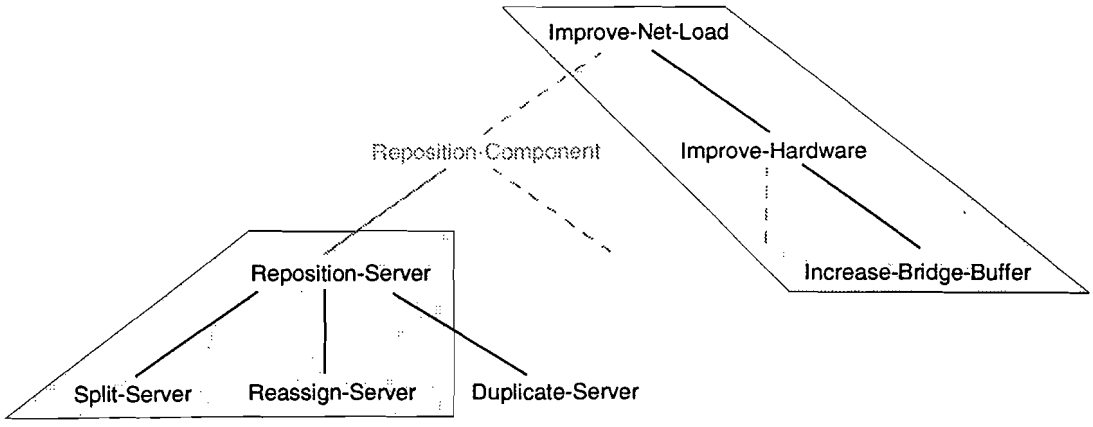



**Fig. 3.** Action hierarchy

We now describe the search process used to find a substitute action. We first note that the set of substitute actions is restricted, because not all actions are suitable. The applicability of an action will be made dependent on certain conditions.

**Definition 10.** An action is further characterized by *preconditions* and *postconditions*. The former give the reason for the application of the action, the latter describe what is achieved by the action.

An action inherits its preconditions and postconditions to its successors in the hierarchy. Each successor action adds specific conditions of its own.

**Definition 11.** An action is *suitable* if its preconditions are satisfied and its postconditions not already realized in the network.

Depending on the values of the preconditions and postconditions of the currently examined action different subspaces of the abstraction hierarchy are searched. Each of these search spaces is traversed in depth-first manner, but applies different exit conditions. Two of these subspaces are shown in fig. 3.

In the example problem situation the solution proposed by the case is:

duplicate-server(Server-1, DataSeg-1)

Its precondition includes that a group of workstations (user group) must use all services of a particular server:

DataSeg-1 is a user group
DataSeg-1 uses the whole service range {database,mail} of server Server-1 in another subnet

Assuming that from the network description in the problem situation we know that Server-1 also supports other services, e.g. a mail service, that is not exclusively used by DataSeg-1. Therefore the duplication of the server would not attack the problem in a consistent way; a substitute action has to be found.

The next action tested is reposition-server, which is more general. The condition about the use of the whole service range is relaxed, now only a group of users must exist that requests services in another subnet. Since the postcondition that there already is a server with these services in the subnet is not satisfied, the search again becomes more specific.

Finally, the action split-server is found. It is suitable because it prescribes to move only those services into the subnet of the user group that are requested by it. It does so by splitting the server into two servers. The application of split-server will make its postcondition true:

All services {database} from {database,mail} of server Server-1 requested by DataSeg-1 are now offered in the same subnet as DataSeg-1

Parameter adaptation is used to adapt the variable components of an action. It comprises component substitution, e.g. the service print-service of Server-2 is substituted by the service database of Server-1, as well as the creation of new parameters, e.g. the action split-server introduces two new parameters that did not exist in duplicate-server:

split-server(Server-1, {mail}, {database}, DataSeg-1)

# 5 Conclusion

The contributions of this paper are twofold. First, it describes a technique for modeling redesign problems in the domain of local area networks. Though this application domain is of great practical relevance, it is also difficult to model. Therefore, there is not much previous work on this topic; one related approach is described in [3]. The notation introduced should be an important step towards documenting redesign situations.

Second, it emphasizes the importance of similarity notions in adaptation. Most research to date has focused on the use of similarity for indexation instead. A good account of this in the redesign context can be found in [1]. We claim that the more noticeable impact of similarity concepts will be on the adaptation of the actions prescribed by a case to the existing problem situation. The adaptation process is based on background knowledge about the suitability of the actions.

The work described in this paper is still in progress. It was done as part of a research project on an environment for concurrent engineering in local area network design [4]. It will contribute a redesign component that can propose design options to the users and should eventually learn new redesign cases from monitoring the designers' activities. The adaptation mechanism has been specified to the level of abstract search algorithms, and a partial analysis of the domain of LAN design has been performed.

# References


1. F. Daube, B. Hayes-Roth, "A case-based mechanical redesign system," in: *Proc. 11th Int. Joint Conf. Artificial Intelligence IJCAI '89*, 1402–1410, 1989.
2. K.J. Hammond, "CHEF," in Ch.K. Riesbeck, R.C. Schank, *Inside Case-based Reasoning.* Lawrence Erlbaum Associates, Hillsdale, New Jersey, 1989.
3. M. Klein. "A computational model for conflict resolution in cooperative design systems," in ed. S.M. Deen, *Cooperating Knowledge Based Systems CKBS-90*, 201–219, Springer-Verlag, 1991.
4. M. Weiss und F. Stetter. "A hierarchical blackboard architecture for distributed AI systems," in ed. G. Tortora, *Proc. 4th Intnl. Conf. Software Engineering and Knowledge Engineering SEKE-92*, Capri, 349–355, IEEE Press, 1992.

# Chapter 3

# Positioning Case-Based Reasoning

# Case-Based and Symbolic Classification Algorithms*
## – A Case Study Using Version Space –

Christoph Globig and Stefan Wess
University of Kaiserslautern, P.O. Box 3049
D-67653 Kaiserslautern, Germany
(globig,wess)@informatik.uni-kl.de

**Abstract**

In the recent years methods of case-based classification were often used in domains traditionally dominated by symbolic learning algorithms. This arises the question, what are the differences and the learning power of the two paradigms. In this work we study the relationship between the case base, the measure of similarity, and the target concept of a case-based learning process. To do so we transform a simple symbolic learning algorithm (the Version Space [10]) in an equivalent case-based variant. The proposed results underline the equivalence of symbolic an case-based methods and show the strong dependency between the measure used by a case-based algorithm and the target concept.

## 1 Introduction

Machine Learning is one of the main research areas in AI. In the recent years this area has branched off significantly. In face of these different approaches arises the question, what are the differencies and what the commonalities of the paradigms. In this work, we want to compare two important inductive learning paradigms – the symbolic and the case-based approach. The learning task we want to study is the classification of objects. The aim of the classification is to map the objects of a universe to concepts, i.e. subsets of the universe. In the most simple scenario the aim is to decide the membership problem of a certain concept, i.e. the universe is separated in two disjunct subsets.

We call the phase during the systems produce their hypothesis *learning phase* and the phase while the hypothesis are used to classify new objects *application phase*. The fundamental problem of both approaches during the learning phase is the same. At every moment the learner knows the correct classification of a finite subset of the universe only. The knowledge that it is able to use to produce a hypothesis is incomplete and therefore the hypothesis need not to be correct. The main difference between the two approaches is the way the learning algorithm produces and represents its hypothesis.

In the learning phase, a *symbolic* algorithm like Version Space [10] builds up a set of rules or a decision tree. In the application phase, these rules are used for the classification of new objects. In contrast, a case-based classifier consists of a finite set of already classified objects – the case base – and a measure of similarity.[1] Given a new object, the system searches in the case base for the nearest neighbor (or the most similar case) and states its classification as the classification of the new object [15]. Learning and the employment of the learned knowledge are not separated but highly integrated. From the viewpoint of machine learning, case-based reasoning (case-based learning) may be seen as a *concept formation task* [3, 14]. If the case-based classifier gets more and more cases, it builds a sequence of pairs $(CB_j, sim_j)$. The aim is to get in the limit a pair $(CB, sim)$, which is a correct classifier for the target concept. This raises the question how the concepts are represented in the systems. Contrary to symbolic learning systems that represent a learned concept *explicitly* by a symbolic formula, case-based systems describe concepts *implicitly* by a pair $(CB, sim)$ [13], i.e. by a measure of similarity $sim$ and a set $CB$ of cases.

The two paradigms of symbolic [7, 8] and case-based classification [2, 1, 4] arise the question which one performs better than the other. In the area of case-based reasoning there is only a very few work concerning the relationship between the used measure of similarity and the set of learnable concepts. The results proposed, e.g., by Cost and Salzberg [4] seem to be too optimistic to us. For the area of Inductive Inference, Jantke [6] proved the equality of the learning power of symbolic and case-based classifiers. The proof is based on the learner's ability to adjust the measure of similarity to the given problem. To use

*The presented work was partly supported by the *Deutsche Forschungsgemeinschaft*, SFB 314: "Artificial Intelligence and Knowledge Based Systems" and the Project IND-CBL.

[1] It is possible to use a distance measure instead of a measure of similarity. We will use the term *measure* to capture both types. For the equivalence of distance and similarity measures see [12].

case-based classification, it is necessary to understand the possibilities to adjust the measure of similarity or, more generally, to understand the use of information in the algorithms [13]. In this paper, we want to show that it is possible to state some relationships and the consequences that follow from this analysis. For our example we have found a direct transformation so that the symbolic and the case-based approach result in an equal classification behavior. We conclude that there is a set of learnable concepts associated with each measure of similarity.

# 2 Example: Version Space

To illustrate the possibility to reformulate a symbolic learning algorithm in a case-based manner we first describe a well-known symbolic algorithm and then the way to construct an equivalent case-based variant. The Version Space algorithm is a simple and well-known symbolic learner [10]. Because of its simplicity, it is easy to show a lot of properties, which hold for many other learning systems, where it would be difficult to prove them. First we want to describe the Version Space.

Let $W_i := N_k$, $(i := 1, \ldots, n)$ be sets of values.[2] $U := W_1 \times \ldots \times W_n$ is the universe. A concept is a vector $K = (K_1, ..., K_n)$, where $K_i = *$ or $K_i = a_i$ $(a_i \in W_i)$. An object $a = (a_1, a_2, \ldots, a_n)$ fulfills the concept $K$ (i.e. $K(a) = 1$), if for all $1 \leq i \leq n$ holds: $K_i \neq *$ or $K_i = a_i$. Otherwise, $K(a)$ is set to 0.

All possible concepts can be arranged in a directed acyclic graph (the Version Space) where the concept at the end of an arrow specifies just one more attribute than the concept at the starting point. The algorithm gets a sequence $(a^i)$ of positive and negative examples of the concept. With respect to the known examples, the Version Space algorithm constructs two sets of concepts. The set $S$ contains all concepts, which are fulfilled by all the known positive and by no known negative example and there is no more specialized[3] concept with the same properties. The set $G$ contains the most general concepts which are fulfilled by all the known positive and by no known negative example.

The algorithm describes the way to modify the sets $S$ and $G$, when a new example is presented. To define $S$ and $G$ properly we force the first example $a^1$ to be positive. The algorithm is based on some assumptions which should be verified. For example, the assertion that the set $S$ has always only one element and that for every concept $g$ from $G$ a more specialized concept with the demanded properties can be found. We do not prove these assertions here. The sets $S$ and $G$ characterize at every moment the classification ability of the Version Space. Under the assumption that the concept is a member of the Version Space every object which fulfills the concept in $S$ must be positive. If the object does not fulfill any concept in $G$ then the object must be negative.

**Version Space algorithm**

1. Initialize $G$ as the set containing only the most general concept
   $G = \{(*, ..., *)\}$ and $S = \{a^1\}$.

2. Assume the new example $a$ is positive.
   Remove all concepts $g$ from $G$ which are not fulfilled by $a$. Search for the most specialized concept $K$ in the Version Space which is fulfilled by all positive examples and set $S = \{K\}$.

   Assume the new example $a$ is negative.
   For every concept $g$ from $G$ which is fulfilled by $a$, search for the most general specializations, which are fulfilled by all known positive and no known negative example. Replace $g$ by the found specializations.

3. If there is a concept $g$ in $G$ which is more specific than a concept in $S$,
   then **HALT**(The examples do not fulfill any concept of the Version Space).

4. If $S = G$
   then **RETURN**(Found concept = $S$)
   else go to 2)

## 2.1 A Case-Based Version Space

It is obvious that the main ability of the Version Space algorithm is to separate relevant and irrelevant values. A value is called *relevant*, if it is part of the concept the learner has to learn. The following

[2] $N_k := \{0, 1, \ldots, k\}$

[3] A concept $K_1$ is called *more specialized* than $K_2$, if $\forall x \in U\ [K_1(x) \Rightarrow K_2(x)] \wedge \exists y \in U\ [K_2(y) \not\Rightarrow K_1(y)]$. The term *more generalized* is defined analogously.

case-based variant (VS–CBR) follows this basic idea. For every attribute $i$ a function $f_i$ is defined that maps the set $W_i$ to $\{0, 1\}$. If the concept $K = (K_1, \ldots, K_n)$ is learned then for every $x_i \in W_i$ holds:

$$f_i(x_i) = \begin{cases} 1 & : \quad K_i = x_i \text{ is possible} \\ 0 & : \quad \text{otherwise} \end{cases}$$

The constructed $f_i$ will be combined to a function $f : U \rightarrow \{0,1\}^n$. The distance between two objects is then defined as

$$d_f(a,b) := |f_1(a_1) - f_1(b_1)| + \ldots + |f_n(a_n) - f_n(b_n)|$$

During the learning phase the function $f$ is learned by the algorithm presented below. It is obvious that every change of the function $f$ will change the distance measure on the universe. Like the original Version Space the first presented case has to be positive to initialize the function $f$.

**Learning Algorithm for $f$**

1. Define $f_i(x_i) = 0$ for all $i$, $x_i \in W_i$
2. If the first positive example is a $= (a_1, a_2, \ldots, a_n)$ define $f_i(a_i) = 1$ for all $i$.
   Define $CB = \{[a, +]\}$
3. Let $b = (b_1, \ldots, b_n)$ be a new example.
   If $b$ is negative, then store $b$ in the case base: $CB := CB \cup \{[b, -]\}$
4. If $b$ is a positive example then for all $i$: If $f_i(b_i) = 0$
   then set $f_i(x_i) = 0$ for all $x_i \in W_i$.
5. If there exists a positive case $p$ and a negative case $n$ in the case base with $d_f(p,n) = 0$ then **HALT**(Not a concept of the Version Space).
6. Delete redundant cases from the case base.[a]
7. If the concept is unequivocal go to step 8) otherwise go to step 3)
8. **RETURN**(The concept is learned)

[a] A case $r$ is redundant if there is a case $s$ in the case base so that $d_f(r,s) = 0$ holds.

Step 5) tests like the symbolic Version Space whether the known examples fit any concept which is learnable. If the learning is done the function $f$ and the case base are used for classification. Given a new object $c$, the set $F := \{fb \mid d_f(fb, c) \leq d_f(fb', c)$ for all $fb'$ in the case base$\}$ is build up. If $F$ contains more than one element the classification is determined by a fixed strategy. For example, the strategy may state the lowest classification value.

## 2.2 Classification with VS–CBR

We want to compare the classification abilities of VS and VS–CBR. In step 2) for all $i$ exactly one $a_i \in W_i$ is mapped to 1. Step 4) occasionally deletes a 1. So, there is never more than one value of an attribute mapped to 1. Let us look at VS and VS–CBR after the presentation of every object. It is obvious that $f_i(a_i) = 1$ holds if and only if the concept in $S$ contains the value $a_i$ for the attribute $i$. The function $d_f$ forces that at every moment $d_f(a,b) = 0$ implies that $a$ and $b$ must be equally classified. Based on these observations, it is easy to verify that objects which can be classified by VS are equally classified by VS–CBR. But VS–CBR will give a classification to every object even if the classification is uncertain. It is possible to suppress this uncertainty by a test of the validity of the classification. If we call the hypothesis when the $i^{th}$ example is presented $VS_i$ or $VS\text{-}CBR_i$, respectively, then $VS\text{-}CBR_i(x) = VS_i(x)$ holds for all $i$ and all $x \in U$.

Positive and negative cases are used differently in VS_CBR during the learning phase.

- Positive cases are used to change $f$, i.e. to adapt the distance measure $d$. They will not be stored in the case base (with the exception of the very first positive example). The distance between any two positive cases is zero.
- Negative cases are stored in the case base but do not change the distance measure.

We have seen that it is possible to rewrite the Version Space algorithm in a case-based manner so that the case-based variant behaves exactly as the symbolic algorithm.

# 3 Basic Issues of Case-Based Classification

In the last paragraph, we have seen a simple case-based classifier. In this paragraph, we want to discuss some basic issues of case-based classifiers and the related learning algorithms. First of all, we have to clarify the conditions which must be fulfilled to learn a concept in a case-based manner. In a second part, we present some examples to show the interdependence between the measure of similarity and the learning power.

A case-based classifier consists of a case base and a measure of similarity (or a distance measure). Neither the case base nor the measure is sufficient for the classification alone. The knowledge about the concept is spread to both. Even in the VS–CBR you can get the concept from the distance measure only because you know the way in which the measure was constructed. If we try to symbolize the relationship we can describe a case base system as a "sum".

**Concept = Case Base + Measure of Similarity**

There are always multiple concepts which can be learned by a given measure. Because of the distribution of the knowledge between the case base and the distance measure it is clear that there are many tuples $(CB, sim)$ which represent the same concept. If the hypothesis of the learner must be modified there are always two possibilities. Either to change the case base or to change the distance measure, *cf.* [9, 5]. VS–CBR uses the positive cases to change the distance measure (by updating the functions $f_i$) while the negative cases are stored in the case base without changing the distance measure.

## 3.1 Simplified Quantitative Analysis

To illustrate the relationship between a case base and a distance measure we simplify the framework for a moment.

1. Let $U$ be a finite universe
2. $d(a,b) = 0 \Rightarrow \forall x \in U[d(x,a) = d(x,b)]$
3. $d$ is fixed.

The assumption 2 means that the relation $\sim$ defined as $x \sim y :\Leftrightarrow d(x,y) = 0$ is an equivalence relation. $\sim$ builds $|U/\sim|$ equivalence classes. It is clear that a concept $K$ is learnable by a measure $d$ if and only if for all $x$, $y \in U$ $d(x,y) = 0 \Rightarrow K(x) \equiv K(y)$ holds, i.e. all elements of an equivalence class must have the same classification. On the other hand, the equivalence classes can be classified without any respect to each other. Therefore, we can conclude that $d$ is able to distinguish between $2^{|U/\sim|}$ different concepts. Each of these concepts can be learned by a case base with $|U/\sim|$ (appropriate) cases (i.e. one case in every equivalence class). As a result, we can state that for the learnability of a concept the only question is the definition of the distance 0. If we have two measures $d$ and $d'$ where $d(x,y) = 0 \Leftrightarrow d'(x,y) = 0$ and $d(x,y) \neq 0 \Leftrightarrow d'(x,y) = 1$ they can recognize the same concepts.

Case based systems can be compared with respect to two important dimensions. The first dimension relates to the implicit knowledge in the measure.

**Definition 1** *A case-based system $(CB_1, sim_1)$ is called to be better informed than a system $(CB_2, sim_2)$ iff they can recognize the same concept and $|CB_1| < |CB_2|$ and, for $i \in \{1,2\}$, there is no $CB_i' \subset CB_i$ so that $(CB_i, sim_i)$ is a classifier of the concept.*

The second dimension relates to the set of learnable concepts.

**Definition 2** *A similarity measure $sim_1$ is called to be more universal than a similarity measure $sim_2$ iff the set of concepts which are learnable by $sim_2$ is a proper subset of the set of concepts which are learnable by $sim_1$.*

To use a universal similarity measure struggles against a minimal case base. To minimize the size of the case base results normally in a less universal similarity measure. We illustrate the countercurrency in figure 1. It lists different distance measures together with the minimal size of the case base to select a certain concept and the total number of learnable concepts. For the table, we use a universe with objects which consists of four attributes. Each attribute can take one value out of 16. So, the size of the universe is 65536. The concept which the measures try to learn fixes two attributes.
We can distinguish two extrem measures:

**Identity of objects:** The similarity is maximal if and only if the compared objects are identical. The measure is universal because it is able to recognize every binary concept in the given universe. But to do so it needs the whole universe as a case base.

**Identity of classification:** The similarity is maximal if the classification of the compared objects is identical. Nearly the whole knowledge about the concept is coded in the measure by the definition of the concept. The case base is used only to exclude some trivial concepts. $sim(x, y) = K(x) \Leftrightarrow K(y)$ can only distinguish four concepts ($K$, $not(K)$, $TRUE$ – i.e. all objects are positive, $FALSE$ – i.e. all objects are negative).

The other measures in the table are between the extremes. VS_CBR$_1$ and VS_CBR$_2$ are neither maximally universal nor able to recognize a concept with a minimal case base. VS_CBR$_1$ is the distance measure, which is built for VS–CBR when the first case is presented. In every dimension exactly one value is mapped to one so that the universe is mapped into the edges of a four dimensional cube. VS_CBR$_2$ is the measure, which is used, when VS–CBR has recognized the concept. It distinguishes only between the two relevant values of the concept and, therefore, builds up only four equivalent classes.

| measure | minimal size of CB | number of learnable concepts |
|---|---|---|
| $sim(x, y) := (x = y)$ | $65536 = 16^4$ | $2^{65536}$ |
| VS_CBR$_1$ | 16 | $65536 = 2^{16}$ |
| VS_CBR$_2$ | 3 (4) | $16 = 2^4$ |
| $sim(x, y) := K(x) \equiv K(y)$ | 2 | $2^2$ |

Figure 1: Measures together with the minimal case base and the number of learnable concepts

The table indicates to describe the relationship between the distance measure and the concept in a different manner: The distance measure determines the space of possible target concepts and the case base selects one of them. In other words we can say that the choice of the distance measure is the bias of case-based classification. Its choice determines the set of target concepts which can be recognized and the efficiency of the learning process as we will see in the next section.

In a typical case-based learner two processes – reducing the size of the hypothesis space and increasing the size of the case base – are done in parallel. So, it is normally difficult to specify the influence of a single case.

The last measure in figure 1 indicates a simple way to rewrite any symbolic algorithm as a case-based one. Use the actual symbolic hypothesis to construct such a measure and store one positive and one negative case in the case base.

# 4 Consequences for Case-Based Classification

We have analyzed the relationship between the measure of similarity, the case base, and the target concept in the described scenario. In the scenario, the learner needs strong preassumptions about the target concept to solve its task with an acceptable number of cases. Preassumptions exclude certain concepts from the hypothesis space. A case-based learner can code this preassumptions in the measure of similarity. Symbolic learners restrict the language to represent their hypothesises.

If we agree to the assertion that there is no measure which depends only on the universe and not on the set of target concepts we must conclude that we are confronted with a bias in case-based classification, too. The bias is the distance measure. Like the bias in symbolic classification, the distance measure determines which concepts are learnable and, in addition, the efficiency of the learning process.

Rendell [11] divides the abstraction done in a learning system in two parts, the bias (to describe the amount of preassumptions) and the power of the learner. We have characterized the learning systems by the number of learnable concepts and the number of cases they need to identify a target concept. The bias relates to the restriction of the set of learnable concepts and is therefore comparable to the degree of universality. The minimal size of the case base reflects the information the learner needs to come to a correct hypothesis. This amount of information is measured by Rendell in the *information gain*. It is therefore very important to select an appropriate distance measure according to the given problem. The measure must have equivalence classes which cover the target concepts and meaningful distance values, i.e. a short distance between two objects must indicate a high probability that the objects have the same

classification. If we know that there is a distance measure with these properties, case-based classification seems to be a good choice. Given an appropriate distance measure, case-based classification has some other useful features. If there is some noise in the data and the effect of the noise is small according to the distance measure then case-based reasoning is a very natural way to implement a noise tolerant learner. In contrast to the results of [4] and [3] we state that the intelligibleness of solutions of a case-based system depends on the intelligibleness of the measure of similarity and is therefore not a property of the case-based approach itself.

To summarize we can say that there is no fundamental advantage or disadvantage of case-based classification [4] compared to the traditional symbolic approach in the simple framework we have considered here. So the question which algorithm is better for a given task depends on the simplicity and adequacy of the representation of the given knowledge. Both approaches need a method to cut down the size of the hypothesis space. While the symbolic approach uses this extraevidential knowledge to construct useful abstractions, the case-based algorithms need it to get appropriate measures of similarity.

# 5 Acknowledgement

The authors thank Michael M. Richter, Klaus P. Jantke, Klaus-Dieter Althoff and the whole research group in Kaiserslautern for many helpful discussions.

# References

[1] David W. Aha. Case-Based Learning Algorithms. In Ray Bareiss, editor, *Proceedings: Case-Based Reasoning Workshop*, San Mateo, California, 1991. DARPA, Morgan Kaufmann Publishers. Washington, D.C., USA, May 8–10, 1991.

[2] David W. Aha, Dennis Kibler, and Marc K. Albert. Instance-Based Learning Algorithms. *Machine Learning*, 6:37–66, 1991. March 1991.

[3] D.W. Aha and D. Kibler. Noise-Tolerant Instance-Based Learning Algorithms. In *Proceedings of the 11th International Conference on Artificial Intelligence IJCAI-89*, pages 794–799. IJCAI, 1989. Detroit, Michigan, USA.

[4] S. Cost and S. Salzberg. A weighted nearest neighbor algorithm for learning with symbolic features. *Machine Learning*, 10(1):56–78, 1993.

[5] Robert S. Holte. Commentary on: Protos an exemplar-based learning apprentice. In Yves Kodtratoff and Ryszard Michalski, editors, *Machine Learning: An Artificial Inteligence Approach*, volume III, pages 128–139. Morgan Kaufmann, 1990.

[6] Klaus P. Jantke. Case-Based Learning in Inductive Inference. In *Proc. COLT-92*, 1992.

[7] R. Michalski, J. G. Carbonell, and T. Mitchell, editors. *Machine Learning: An Artificial Intelligence Approach*, volume 1. Tioga, Palo Alto, California, 1983.

[8] R. Michalski, J. G. Carbonell, and T. Mitchell, editors. *Machine Learning: An Artificial Intelligence Approach*, volume 2. Morgan Kaufmann, Palo Alto, California, 1986.

[9] Ryszard S. Michalski. Concept Meaning, Matching and Cohesiveness. In Stella Vosniadou and Andrew Ortony, editors, *Similarity and Analogical Reasoning*, chapter 4, pages 122–145. Cambridge University Press, Cambridge, 1989.

[10] T.M. Mitchell. Gerneralization as search. *Artificial Intelligence*, 18(2):203–226, 1982.

[11] L. Rendell. A General Framework for Induction and a Study of Selective Induction. *Machine Learning*, 1:177–226, 1986.

[12] M. M. Richter and S. Wess. Similarity, Uncertainty and Case Based Reasoning in PATDEX. In Robert S. Boyer, editor, *Automated Reasoning - Essays in Honor of Woody Bledsoe*, pages 249–265. Kluwer Academic Publishers, 1991.

[13] Michael M. Richter. Classification and learning of similarity measures. In *Proc. of the 16th Annual Conference of the German Society for Classification (Gesellschaft für Klassifikation e.V.)*. Springer Verlag, 1992.

[14] Steven Salzberg. Distance Metrics for Instance-Based Learning. In Z. W. Ras and M. Zemankova, editors, *Proceedings of the 6th International Symposium on Methodologies for Intelligent Systems (ISMIS'91)*, volume 542 of *Lecture Notes in Artificial Intelligence*, pages 399–408, Berlin, 1991. Springer-Verlag. Charlotte, North Carolina, USA, October 1991.

[15] Craig Stanfill and David Waltz. The Memory Based Reasoning Paradigm. In Janet L. Kolodner, editor, *Proceedings Case-Based Reasoning Workshop*, pages 414–424, San Mateo, California, 1988. DARPA, Morgan Kaufmann Publishers. Clearwater Beach, Florida, USA, May 10–13, 1988.

# Case-Based Representation and Learning of Pattern Languages

(extended abstract)

Klaus P. Jantke and Steffen Lange *

Hochschule für Technik, Wirtschaft und Kultur Leipzig (FH)
Fachbereich Informatik, Mathematik & Naturwissenschaften
P.O.Box 66, 04251 Leipzig, Germany

**Abstract**

Pattern languages seem to suit case-based reasoning particularly well. Therefore, the problem of inductively learning pattern languages is paraphrased in a case-based manner. A careful investigation requires a formal semantics for case bases together with similarity measures in terms of formal languages. Two basic semantics are introduced and investigated. It turns out that representability problems are major obstacles for case-based learnability. Restricting the attention to so-called proper patterns avoids these representability problems. A couple of learnability results for proper pattern languages are derived both for case-based learning from only positive data and for case-based learning from positive and negative data. These result exhibit the importance of flexible non-standard approaches to similarity. The chosen semantics determine which type of similarity measure support representability and learnability.

*Was sich überhaupt sagen läßt, läßt sich klar sagen; und wovon man nicht reden kann, darüber muß man schweigen.*

LUDWIG WITTGENSTEIN
Tractatus Logico-Philosophicus, 1922

## 1 Motivation

Case-based reasoning is a currently booming area in artificial intelligence. Research papers are mushrooming, thus, providing a huge amount of cases for case-based reasoning. As outsiders, we became interested in the area, as cases seem to play a role particularly similar to the role of examples in our work in inductive inference. We found it extremely difficult to make this first rough observation more precise. One crucial reason is the lack of formalization in a large number of case-based reasoning approaches. Thus, we decided to approach our problem by paraphrasing inductive inference in terms of case-based learning in an area which seems particularly tailored to fit the gist of case-based reasoning. This is the area of pattern languages, more precisely, the domain of learning pattern languages from positive or both positive and negative cases. In this well-formalized research area, we did some investigations focussing on clear results valid under clear assumptions. For example, we tried to find out how particular semantics influence the type of similarity measures suitable for succesful learning. We could prove with mathematical precision that the symmetry of similarity concepts is rarely desirable, e.g. Interestingly, our results may be interpreted in case-based reasoning and are throwing some light on essential problems of case-based reasoning, in general.

We are interested in results of *mathematical precision* exhibiting fundamental phenomena related to case-based reasoning. Although our work presented has been mainly driven by the learnability investigations reported in chapter 4, we consider the results of chapter 3 as basic.

*The work has been partially supported by the German Federal Ministry for Research and Technology (BMFT) within the Joint Project (BMFT-Verbundprojekt) **GOSLER** on **Algorithmic Learning for Knowledge-Based Systems** under contract no. 413-4001-01 IW 101 A and by the DFG-Project **IND-CBL** under reference Ja 566/2-1.

# 2 Introduction

This paper is dealt with problems of case-based learning in a particular area where we can exploit a remarkable amount of inductive learning results. This is the area of pattern languages as introduced in [3]. This area has attracted enormous attention in learning theory (cf. [3], [17], [16], [6], [15], [7], [14], and others). A key reason for the intensive research work dedicated to the learning of pattern languages is the naturalness of the general learning problem as well as the closeness of individual texts to the general underlying pattern structures. From this insight, there is outgrowing a particular motivation of the investigations presented here.

Here, we are briefly illustrating what will be considered in more detail below. Due to the very restricted space, we can not present any more detailed technical discussion or proof[1]. Instead, we put more emphasis on illustrations. Given any text structure like

$$x_{author}, x_{title}, x_{journal}\, x_{volume}\, (x_{year}), x_{pages}$$

one may easily imagine a number of typical instances. Vice versa, from some typical cases like

> Dana Angluin and Carl H. Smith, A Survey of Inductive Inference: Theory and Methods, Computing Surveys 15 (1983), 237-269
>
> Reinhard Klette and Rolf Wiehagen, Research in the Theory of Inductive Inference by GDR Mathematicians - A Survey, Information Sciences 22 (1980), 149-169

most people will infer underlying patterns like the one above. In this particular domain, there is an easy concept of cases, and humans are usually able to learn from a small number of those cases (cf. [16] for experiments and measurements on automated pattern inference).

This consideration motivated the following intention. First, if pattern inference is an area where we have a natural and easy to understand concept of cases, we should be able to develop and illustrate basic ideas of case-based learning. Second, if there are general difficulties of case-based learning in such a nice area, this could be understood as testbed for problems we are faced to in a large number of areas where formal considerations may be of a considerably greater complexity. In a sense, the results about case-based learning of pattern languages developed in the sequel may be interpreted as *lower bounds* for the difficulties of case-based learning in a huge variety of further areas.

## 2.1 Case-Based Learning

Case-based reasoning is a recently booming subarea of artificial intelligence. One important reason is that human experts tend to use knowledge in the form of particular cases or episodes rather frequently than generalized knowledge as described by rules, e.g. Therefore, there is some hope that case-based reasoning may help to widen the bottleneck of knowledge acquisition. The reader is directed to [13] for a recent introduction in and survey of case-based reasoning. Within case-based reasoning, case-based learning as investigated in [1] and [2], for instance, is a rather natural way of designing learning procedures. Recent formalizations (cf. [8]) have exhibited the remarkable power of case-based learning algorithms.

## 2.2 Text Patterns

Following [3], a pattern is a non-empty string build over some alphabet $A$ and some disjoint set of variables $X = \{x_1, x_2, \ldots\}$. By $\mathcal{P}$ we denote the set of all patterns, i.e. $\mathcal{P} = (A \cup X)^+$. $\mathcal{PP} = \mathcal{P} \setminus A^+$ denotes the set of so-called proper patterns. For a pattern $p$, we denote by $\mathcal{L}(p)$ the corresponding pattern language defined by $p$. $\mathcal{L}(p)$ contains all strings which can be obtained by substituting non-empty strings for the variables of $p$, where the same variables have to be substituted by the same strings.

Pattern languages form the basis of a couple of applications in different fields, e.g. in the intelligent text processing system EBE (cf. [16]) or in a classification system for transmembrane proteins (cf. [5]).

## 2.3 Inductive Pattern Inference

Inductive inference is the process of hypothesizing a general rule from eventually incomplete data. It has its origins in philosophy of sciences. During the last three decades, it received much attention in computer science (cf. [4]).

The general situation investigated in language learning can be described as follows: There is some target language to be learnt (identified, ...) inductively. Given more and more possibly incomplete information concerning the language to be learnt, an inference device has to produce in every step a hypothesis about

[1] For the proofs, the reader may consult [10]. A version of this paper focussed to learning problems appears as [11].

the phenomenon to be inferred. The set of all admissible hypotheses is called space of hypotheses. The given information may contain only *positive examples* (indicated by the suffix $TXT$ below), i.e. exactly all the strings contained in the language to be recognized, or both *positive and negative examples* (indicated by $INF$), i.e. the learner is fed with arbitrary strings over the underlying alphabet which are classified with respect to their containment to the unknown language. The sequence of hypotheses has to converge to a hypothesis correctly describing the object to be learnt. To sum up, the inference process as a whole is a limiting one.

Our learnability concept (cf. [10], [11]) is an immediate adaptation of the classical identification types in recursion-theoretic inductive inference (cf. [4], [12]). It is reflecting the approaches underlying [3], [17], [15], e.g. Using standard notations, the following learnability results may be assumed.

**Theorem 1**
(1) $\mathcal{P} \in LIM.TXT$
(2) $\mathcal{P} \in LIM.INF$

# 3 Case-Based Representation of Pattern Languages

If some algorithm is expected to learn any member of some class of objects in a case-based manner by processing information about particular target objects to come up with some finite case-base and some similarity measure describing the particular target object, this obviously assumes some interpretation of pairs built from case-bases and similarity functions in terms of the objects under consideration. Formally spoken, one needs some well-defined semantics. In general, there is no standard semantics. [8] is introducing three slightly different semantics, in a particular setting. Similarly, the reader will find below two slightly different approaches used in the paper on hand. It is especially surprising that a remarkable number of papers do not make the chosen semantics explicit. But for a formally correct treatment, the choice of some precise semantics is inevitable. The reader may check our theorems below and their proofs in this regard (cf. [10]).

## 3.1 Semantics

There is assumed some finite, non-empty alphabet $A$. Cases about some formal language are labelled words indicating whether or not some word provided belongs to the language to be represented or even to be learnt. For labelling words, we choose 0 and 1 meaning *no* and *yes*, respectively. Certain papers in the area of case-based reasoning provide some rough concept of semantics as follows (cf. [1], [2], for example). If there is some finite case-base $CB$ and some given similarity measure $\sigma$, this classifies words $w$ according to the following pcocedure: Search $CB$ for some labelled word $(v, d)$ where $\sigma(v, w)$ is maximal. Return $d$ to classify $w$. There may obviously arise some ambiguity, if there are conflicting classifications by cases $(v_1, 0)$ and $(v_2, 1)$ where both $v_1$ and $v_2$ are of maximal similarity to $w$. There are several ways to resolve those conflicts. Two of them are chosen for the formal semantics introduced in the sequel. The *standard* approach and the *competing* approach will be denoted by $\mathcal{L}_{st}(CB, \sigma)$ and $\mathcal{L}_{c}(CB, \sigma)$, respectively.

Any formal semantics has to be based on some similarity concept. Therefore, before specifying the intended semantics, we have to put some emphasis on similarity.

### 3.1.1 Similarity Concepts

The majority of current publications in case-based reasoning is considering cases as tuples over some chosen collection of attributes. For every attribute $a_i$, there is some domain $D_i$ of possible attribute values. Usually, $D_i$ is equipped with some metric $\delta_i$ to describe the *distance* of any two corresponding attribute values. This allows to express the distance of two tuples $t_1$ and $t_2$ by a HAMMING distance $\delta$ by $\delta(t_1, t_2) = \sum_{i=1}^{n} w_i \cdot \delta(t_1.a_i, t_2.a_i)$. Usually, distances are transformed to describe *similarities*. There is a standard way which seems to be used in most approaches: $\sigma(t_1, t_2) = 1 - \frac{\delta(t_1,t_2)}{1+\delta(t_1,t_2)}$. The richness of current problems attacked by case-based reasoning approaches bears abundant evidence of the need of more sophisticated similarity concepts. [9] is intended to be one step towards *structural similarity concepts*. Throughout the present paper, we are not going to invoke structural approaches to similarity. But we are interested in more flexibility than provided by encoded HAMMING distances.

For the purpose of the present extended abstract, $\Sigma$ is chosen to denote the class of all total recursive similarity concepts (cf, [10]). Elements $\sigma$ of $\Sigma$ may be either $0,1$-valued or mapping into the rational numbers ranging from 0 to 1.

### 3.1.2 Formal Semantics

There may be further approaches to the semantics of case bases together with similarity functions for formal languages, but the two considered seem to be basic. Assume $CB \subseteq A^+ \times \{0,1\}$ and $\sigma \in \Sigma$ as introduced above.

**Definition 1**

$\mathcal{L}_{st}(CB,\sigma) = \{w/\exists(u,1) \in CB\,[\sigma(u,w) > 0 \wedge \forall(v,0) \in CB\,[\sigma(u,w) > \sigma(v,w)]]\}$

$\mathcal{L}_{c}(CB,\sigma) = \{w/\exists(u,1) \in CB\,[\sigma(u,w) > 0 \wedge \forall(v,d) \in CB\,[u \neq v \Rightarrow \sigma(u,w) > \sigma(v,w)]]\}$

The presentation of the following lemmata has a twofold intention. First, these lemmata provide some insight into the nature of the semantics considered. They illustrate both common features and differences of these semantics. Second, these lemmata provide a firm basis for understanding and proving the following theorems (cf [10] for details).

**Lemma 1**

$\forall\sigma \in \Sigma\forall p \in \mathcal{P}\forall CB \subset_{fin} L(p) \times \{1\} \cup \overline{L(p)} \times \{0\}$

$(\ |CB \cap L(p) \times \{1\}| = 1 \implies \mathcal{L}_c(CB,\sigma) = \mathcal{L}_{st}(CB,\sigma)\ )$

**Lemma 2**

$\forall\sigma \in \Sigma\forall X \subseteq A^+ \times \{0,1\}\forall Z \subseteq A^+ \times \{1\}(\ \mathcal{L}_{st}(X,\sigma) \subseteq \mathcal{L}_{st}(X \cup Z,\sigma)\ )$

**Lemma 3**

$\forall\sigma \in \Sigma\ (\ \sigma\ idempotent \implies \forall X \subseteq A^+ \times \{1\}\ (\ X \subseteq \mathcal{L}_{st}(X,\sigma) \times \{1,0\}\ )\ )$

**Lemma 4**

$\forall\sigma \in \Sigma\forall p \in \mathcal{P}\forall CB \subset_{fin} L(p) \times \{1\} \cup \overline{L(p)} \times \{0\}$

$(\ \mathcal{L}_c(CB,\sigma) \subseteq \mathcal{L}_{st}(CB,\sigma)\ )$

**Lemma 5**

$\forall\sigma \in \Sigma\forall p \in \mathcal{P}\forall CB \subset_{fin} L(p) \times \{1\} \cup \overline{L(p)} \times \{0\}$

$\exists\sigma' \in \Sigma\exists CB' \subset_{fin} L(p) \times \{1\} \cup \overline{L(p)} \times \{0\}\ (\mathcal{L}_{st}(CB,\sigma) = \mathcal{L}_c(CB',\sigma'))$

**Lemma 6**

$\forall\sigma \in \Sigma\forall p \in \mathcal{P}\forall CB \subset_{fin} L(p) \times \{1\} \cup \overline{L(p)} \times \{0\}$

$\exists\sigma' \in \Sigma\exists CB' \subset_{fin} L(p) \times \{1\} \cup \overline{L(p)} \times \{0\}\ (\mathcal{L}_c(CB,\sigma) = \mathcal{L}_{st}(CB',\sigma'))$

$\sigma$ is called *idempotent*, if $\sigma(x,x) = 1$ holds for all admissible arguments $x$. This property is deemed important, but the proofs of our theorems below (cf. [10]) show that it can rarely be achieved.

The lemmata show that both semantics, although they have the same expressive power, behave differently in some respect. This will be used below.

## 3.2 Representability Results

In the results listed below, the notation $\mathcal{L}_*$ refers to both the standard semantics and the competing semantics as introduced above. For the readers convenience, every theorem will be paraphrased (in italics), first.

*Under both semantics, there is no universal similarity measure $\sigma$ which allows to represent every pattern language by a finite number of its elements considered as positive cases.*

**Theorem 2**

$\neg\exists\sigma \in \Sigma\forall p \in \mathcal{P}\exists CB \subset_{fin} \mathcal{L}(p) \times \{1\}\,(\mathcal{L}(p) = \mathcal{L}_*(CB,\sigma))$

*Under standard semantics, there is some universal similarity measure $\sigma$ which allows to represent every pattern language by a finite number of cases, where the words of these cases are not restricted to be taken from the target language itself.*

**Theorem 3**

$\exists\sigma \in \Sigma \forall p \in \mathcal{PP} \exists CB \subset_{fin} A^+ \times \{1\} (\mathcal{L}(p) = \mathcal{L}_*(CB, \sigma))$

*Under both semantics, there is a universal similarity measure $\sigma$ which allows to represent every proper pattern language by a finite number of its words considered as positive cases.*

**Theorem 4**

$\exists\sigma \in \Sigma \forall p \in \mathcal{PP} \exists CB \subset_{fin} \mathcal{L}(p) \times \{1\} (\mathcal{L}(p) = \mathcal{L}_*(CB, \sigma))$

*Under both semantics, there is no universal similarity measure $\sigma$ being symmetric which allows to represent every proper pattern language by a finite number of its words considered as positive cases.*

**Theorem 5**

$\neg\exists\sigma \in \Sigma [\sigma \text{ symmetric} \wedge \forall p \in \mathcal{PP} \exists CB \subset_{fin} \mathcal{L}(p) \times \{1\} [\mathcal{L}(p) = \mathcal{L}_*(CB, \sigma)]]$

*Under standard semantics as well as under competing semantics, there is a universal similarity measure $\sigma$ allowing to represent every pattern language by a finite case-base $CB$ of both examples and counter-examples considered as positive and negative cases, respectively.*

**Theorem 6**

$\exists\sigma \in \Sigma \forall p \in \mathcal{PP} \exists CB \subset_{fin} \mathcal{L}(p) \times \{1\} \cup \overline{\mathcal{L}(p)} \times \{0\} (\mathcal{L}(p) = \mathcal{L}_*(CB, \sigma))$

At the very moment, it is still open whether or not **Theorem 6** is valid, if it is required that the corresponding similarity measure $\sigma$ is symmetric. We conjecture that there does not exist any symmetric similarity measure $\sigma$ which allows to represent the class of all pattern languages using positive and negative cases under any of the two semantics investigated.

# 4 Case-Based Learning of Pattern Languages

Because of the lack of space, we can provide a list of annotated results, only.

## 4.1 Learning Scenario

All the formalisms may be found in [11]. It is sufficient to understand the basic scenario. Some pattern language is learnable from text or informant under one of the semantics introduced, if there is a universal learning device able to collect cases from any text or informant, respectively, such that it is collecting a case base in the limit, which is only finite, and which describes the target language correctly under the assumed semantics. Any similarity concept is assumed.

## 4.2 Learnability Results

**Theorem 4** and **Theorem 6** circumscribe the possibilities of case-based learning of pattern languages. Again, every theorem will be paraphrased for the readers convenience.

*For the class of proper pattern languages, there are universal case-based learning algorithms based on text for both semantics considered.*

**Theorem 7**
(1) $\mathcal{PP} \in S - CBL.TXT$
(2) $\mathcal{PP} \in C - CBL.TXT$

**Corollary**

(1) $\mathcal{PP} \in S - CBL.INF$
(2) $\mathcal{PP} \in C - CBL.INF$

*Under competing semantics, the whole class of pattern languages $\mathcal{P}$ is case-based learnable from positive and negative examples.*

**Theorem 8**
$\mathcal{P} \in C - CBL.INF$

# 5 Discussion

The results above and their corresponding proofs are raising a considerable number of questions about the interplay of semantics, types of similarity measures, and learnability concepts. Because of the lack of space, we can mention only three of them:

- The proofs of the **Theorems 3**, **4**, and **6** invoke similarity measures which do not meet human intuition quite well. What is the expressiveness of similarity concepts reflecting certain human ideas of similarity of strings under particular semantics?
- If one chooses standard semantics in **Theorem 7**, $\{0,1\}$-valued similarity measures are sufficiently expressive. This does not hold under competing semantics. How to characterize similarity concepts which admit $\{0,1\}$-valued similarity measures?
- Until now, it is still open whether a result similar to **Theorem 8** can be achieved under standard semantics, too.

# References


[1] David W. Aha. Case-based learning algorithms. In *DARPA Workshop on Case Based Reasoning*, pages 147–157. Morgan Kaufmann, 1991.

[2] David W. Aha, Dennis Kibler, and Marc K. Albert. Instance-based learning algorithms. *Machine Learning*, 6:37–66, 1991.

[3] Dana Angluin. Finding patterns common to a set of strings. *Journal of Computer and System Sciences*, 21:46–62, 1980.

[4] Dana Angluin and Carl H. Smith. A survey of inductive inference: Theory and methods. *Computing Surveys*, 15:237–269, 1983.

[5] Setsuo Arikawa, Satoru Kuhara, Satoru Miyano, Yasuhito Mukouchi, Ayumi Shinohara, and Takeshi Shinohara. A machine discovery from amino acid sequences by decision trees over regular patterns. In *Intern. Conference on Fifth Generation Computer Systems, June 1-5, 1992*, volume 2, pages 618–625. Institute for New Generation Computer Technology (ICOT), Tokyo, Japan, 1992.

[6] Klaus P. Jantke. Polynomial time inference of general pattern languages. In M. Fontet/ K. Mehlhorn, editor, *STACS'84*, Lecture Notes in Computer Science 166, pages 314–325. Springer-Verlag, 1984.

[7] Klaus P. Jantke. Monotonic and nonmonotonic inductive inference of functions and patterns. In K.P. Jantke J. Dix and P.H. Schmitt, editors, *Nonmonotonic and Inductive Logic, 1st Int. Workshop*, Lecture Notes in Artificial Intelligence 543, pages 161–177. Springer-Verlag, 1991.

[8] Klaus P. Jantke. Case based learning in inductive inference. In *Proc. of the 5th ACM Workshop on Computational Learning Theory, (COLT'92), July 27-29, 1992, Pittsburgh, PA, USA*, pages 218–223. ACM Press, 1992.

[9] Klaus P. Jantke. Types of incremental learning. In *Working Notes, AAAI Symposium on Training Issues in Incremental Learning, March 23-25, 1993, Stanford, CA, USA*, pages 26–32. Stanford University, 1993.

[10] Klaus P. Jantke and Steffen Lange. Case-based represantation and learning of pattern languages. GOSLER Report 17/92, Technische Hochschule Leipzig, FB Mathematik & Informatik, October 1992.

[11] Klaus P. Jantke and Steffen Lange. Case–based representation and learning of pattern languages. In K.P. Jantke, S. Kobayashi, E. Tomita, and T. Yokomori, editors, *4th Intern. Workshop on Algorithmic Learning Theory (ALT'93), Proc., November 1993, Tokyo, Japan*, Lecture Notes in Artificial Intelligence, pages 87–101. Springer–Verlag, 1993.

[12] Reinhard Klette and Rolf Wiehagen. Research in the theory of inductive inference by GDR mathematicians - a survey. *Information Sciences*, 22:149–169, 1980.

[13] Janet L. Kolodner. An introduction to case-based reasoning. *Artificial Intelligence Review*, 6:3–34, 1992.

[14] Steffen Lange. A note on polynomial-time inference of k-variable pattern languages. In K.P. Jantke J. Dix and P.H. Schmitt, editors, *NIL'90, Nonmonotonic and Inductive Logic, 1st Int. Workshop*, volume 543 of *Lecture Notes in Artificial Intelligence*, pages 178–183. Springer–Verlag, 1991.

[15] Steffen Lange and Rolf Wiehagen. Polynomial-time inference of arbitrary pattern languages. *New Generation Computing*, 8(4):361–370, 1991.

[16] Robert P. Nix. Editing by examples. Technical Report 280, Yale University, Dept. Comp. Sci., 1983.

[17] Takeshi Shinohara. Polynomial time inference of pattern languages and its applications. In *7th IBM Symposium on Mathematical Foundations of Computer Science*, pages 191–209, 1982.

# A Comparison of Case–Based Learning to Search–Based and Comprehension–Based Systems

Josef Krems, Josef Nerb, Franz Schmalhofer & Bidjan Tschaitschian

Department of Psychology University of Regensburg
Postfach 397
93053 Regensburg, Germany
e–mail: (josef.krems, nerb)@rpss3.psychologie.uni–regensburg.de

German Research Center for Artificial Intelligence
University Bldg. 57, Erwin–Schroedinger–Strasse, Postfach 2080
67663 Kaiserslautern, Germany
e–mail: (schmalho, tschaits)@dfki.uni–kl.de

**Abstract**

This paper compares the performance of a typical case–based learner to search– and comprehension–based learning systems. Various Tower of Hanoi tasks are used as a testbed for evaluating the strengths and potential shortcomings of the three different learning schemes. The amount and types of knowledge that are required for the successful performance will be identified for each of the three systems. In addition, the performance of the three learning systems is compared to respective data from a psychological experiment.

## 1 Case–, Search– and Comprehension–based Learning

Case–based planners (Hammond, 1989) acquire additional knowledge by storing new cases (i.e. the specific plans for different problems); search–based systems like SOAR or PRODIGY learn by chunking the result of a search process (Rosenbloom et al., 1991a; Minton et al., 1989), by compilation (Anderson, 1987), and by forming macro–operators (Korf, 1985). The chunking mechanism of SOAR, which resembles or is identical to explanation–based generalization, is also capable of learning at the knowledge level (Rosenbloom et al. 1991b). This technique is known as data–chunking. It enables SOAR to memorize declarative structures (e.g. plans or prior problem solving experiences) explicitly which is a necessary ability to model the behavior of case–based systems. This memorizing is conducted by an operator and thus is a knowledge–based and deliberate act. Generally, SOAR productions are better thought of as memory retrievers than as operators or procedures (Rosenbloom et al., 1991a). The mapping from a case–based system onto SOAR as a search–based system is therefore rather straightforward, as was just outlined by Akyurek (1992): Suppose there is a plan for achieving some goal. The chunking mechanism adds this plan which is indexed by its (generalized) goal conditions to the long–term recognition memory. Confronted with a similar task this declarative structure can be retrieved and be used as a template for achieving the task. Plan modification and repairs are both accomplished by the default problem–solving capabilities of SOAR. The resulting new plan again can be stored with the data–chunking mechanism for later recall. The gain for cognitive modeling using this approach is twofold. First, you have to make explicit which part of a present case will be memorized and what can serve as its retrieval cue. Note that beside generalization which results automatically from the chunking mechanism, also abstracted versions of the cases at hand can stored. All necessary knowledge for doing this abstraction process (e.g. prior known concepts) has to be part of the model. Second, you need to have a theory about when this case storing event will happen.

Comprehension–based systems (Mannes & Kintsch, 1991; Wharton & Kintsch, 1991) offer a third possibility for learning: From specific problem solving experiences (cases) and a related problem description (text) some coarse–grained abstract representation is constructed, that may initially be inconsistent and redundant. By holistic integration processes a coherent and consistent procedure schema is subsequently formed. Such a procedure schema can be reused for obtaining solutions to problems which are quite different at the concrete level, but have been comprehended to share abstract commonalities. The

importance of forming and reformulating abstract representations has recently also been pointed out by Clancey (1989).

## 2 Different Tower of Hanoi Problems as a Testbed

The Tower of Hanoi task (Simon, 1975) requires that a tower of n disks which are graded in size is transferred from a start peg (say peg A) to some goal peg (say peg C). At the outset, all the disks are arranged pyramidically on peg A. An additional peg (say peg B) may be used as an auxiliary location. The following rules must be observed: At any time only a single disk may be moved and a larger disk must never be put on top of a smaller disk.

A problem solving system will represent the individual disks in a concrete description language. A problem state of the Tower of Hanoi problem can be represented by a list. The initial states of the 3–, 4–, and 5– disk problems are thus represented by [[1 2 3] [] []], [[1 2 3 4 ] [] []] and [[1 2 3 4 5] [] []] respectively: 1 refers to the smallest disk, 2 to the second smallest disk and so on.

For the 3 different types of learning systems, the acquisition and utilization of knowledge was tested in the following way: Each system (case–, search–, and comprehension–based) was alternatively trained with three different case–data (3–, 4–, and 5– disk problems). The utilization of the acquired knowledge was then tested with the 4–disk problem.

## 3 Practical Results Concerning the Comparison of the 3 Approaches

For the Tower of Hanoi tasks the comparison of the acquisition and utilization of knowledge in case–based, search–based, and comprehension–based systems yields the following results: A case–based planner stores the specific experiences and utilizes these experiences by adapting them to new problems. The specific solutions of 5–disk (odd number of disks) and 4–disk problems (even number of disks) are quite different (Simon, 1975). A case–based system with a 5–disk training required a substantial amount of new knowledge for refitting the solution of the 5–disk problem for solving 4–disk problems. Search–based systems acquire new knowledge by searching a problem space and forming macro–operators or chunks. Since the Tower of Hanoi task results in an ill–suited problem decomposition, macro–operators are formed which yield quite inefficient problem solutions (Korf, 1985). Quite often, these macro–operators cannot be transferred between problems with a different number of disks. When multiple levels of descriptions are available like in SOAR, more useful chunks of problem solving experiences can be formed (Ruiz & Newell, 1989).

A SOAR model which learns chunks as a result of a lookahead–search combined with a goal–decomposition strategy ("move biggest disk not–on–C to C") predicts only positive transfer from a 4–disk training to the 4–disk criterion task. Transfer from an odd–numbered training task to an even–numbered task is negative.

A second SOAR model which uses the cased–based approach described above for memorizing small episodes (cases) during solving the training problems predicts better transfer from a 5–disk training to a 4–disk criterion task due to more opportunities for storing episodes. This model which is influenced by the concurrent protocol analysis recently reported by VanLehn (1991) stores abstracted experiences acquired during the "major moves" which form a stable pattern of $4k + 1$ moves in the protocol. At first the abstraction process uses the prior known concept of a pyramid and then secondly, learns to conceive the size of the pyramid a further relevant concept.

With a comprehension–based approach an abstract procedure schema is formed in terms of situation knowledge. The same abstract schema is thus acquired for all Tower of Hanoi tasks with more than 4 disks. The schema acquired from the 3– and 4–disk problems is still similar but less elaborated. Comprehension–based learning from a 5–disk training therefore even produces better solutions for the 4–disk criterion task than a 4–disk training.

## 4 Empirical Results from a Psychological Experiment

In order to evaluate the psychological validity of case–, search–, and comprehension–based learning, an experiment with human subjects was performed. In three different conditions (30 subjects each), subjects had to solve two 3–disk, two 4–disk, or two 5 disk–problems in a row and their number of moves was recorded. The two consecutive problems were similar. The first time the tower of disks was on peg A in the initial state. The second time it was located on peg B. Both times, it had to be transferred to peg C.

Thereafter, the knowledge which they had acquired from these problem solving episodes was tested. In order to obtain a more complete assessment of their knowledge with respect to the 4–disk problem, which served as the criterion task, all subjects were presented with the 81 different states of the four disk problem, one at a time. Rather than completely solving the Tower of Hanoi problem, the subject had to select the best move for each of the 81 different problem states, which were randomly divided in three sets of 27 states. For selecting a move, subjects were allowed 15, 30 or 45 seconds. The allowed processing time and the three sets of 27 states were counterbalanced by a Latin–Square design.

Tabelle 1. Average number of moves in three different training tasks

| Mean number of moves | 3–disk problem | 4–disk problem | 5–disk problem |
|---|---|---|---|
| first problem | 11.1 | 30.9 | 66.0 |
| second problem | 9.1 | 21.4 | 55.0 |

Table 1 shows the average number of moves for two similar problems with 3, 4 or 5 disks. Clearly, fewer moves were required when solving the problem for the second time as compared to the first time. Table 2 shows the performance in the 4–disk criterion task. The average number of correct moves is shown as a function of the allowed processing time (15, 30, or 50 seconds) and the three types of training (solving 3–, 4–, or 5– disk problems).

Tabelle 2. Percentage of optimal moves in the 4–disk criterion task as a function of pretraining and processing time

| | Pretraining | | |
|---|---|---|---|
| Time (s) | 3–disks | 4–disks | 5–disks |
| 15 | 63 | 68 | 70 |
| 30 | 69 | 75 | 77 |
| 45 | 69 | 76 | 81 |

Independent of the specific processing time which was allowed for selecting the best move from a given state, the subjects with the 5–disk problem solving experience performed better than the subjects with the 4– or 3–disk problem solving experiences. We may thus conclude that the subjects did not solely store the problem solving moves for the specific Tower of Hanoi problem nor solely compiled solution knowledge for it. Instead subjects must have also utilized background knowledge which allowed them to form a more abstract representation from the problem solving episodes, that could be efficiently transferred from the 5–disk training task to the 4–disk test task. Since more elaborate abstractions can be acquired from the 5–disk problem, the subjects with this training performed better in the criterion task than the subjects with the 4– or 3–disk training.

# 5 References


Akyurek, A. (1992). On a computational model of human planning. In J. A. Michon & A. Akyurek (Eds.), *Soar: A cognitive architecture in perspective* (pp. 81–108). Dodrecht, The Netherlands: Kluwer.

Anderson, J. R. (1987). Skill Acquisition: Compilation of weak–method problem solutions.*Psychological Review* 94(2):192–210.

Clancey, W.J. (1989). The knowledge level reinterpreted: Modeling how systems interact *Machine Learning* 4:285–291.

Hammond, K. (1989). *Case–based planning*. London: Academic Press.

Korf, R. E. (1985). Macro–operators: A weak method for learning. *Artificial Intelligence* 26:35–77.

Mannes, S. M., and Kintsch, W. (1991). Routine computing tasks: Planning as understanding. *Cognitive Science* 15:305–342.

Minton, T. M., Carbonell, J. G., Knoblock, C. A., Kuokka, D. R., Etzioni, O., and Gil, Y. (1989). Explanation–based learning: A problem solving perspective. *Artificial Intelligence* 40:63–118.

Rosenbloom, P. S., Laird, J. E., Newell, A., and McCarl, R. (1991a). A preliminary analysis of the SOAR architecture as a basis for general intelligence. *Artificial Intelligence* 47: 289–325.

Rosenbloom, P.S., Newell, A., Laird, J. E. (1991b). Toward the knowledge level in Soar: The role of the architecture in the use of knowledge. In K. VanLehn (Ed.) *Architectures for intelligence*. Hillsdale, NJ: Erlbaum.

Ruiz, D. and Newell, A. (1989). Tower–noticing triggers strategy–change in the Tower of Hanoi: A Soar Model. *Cognitive Science Proceedings*.

Simon, H. A. (1975). The functional equivalence of problem–solving skills. *Cognitive Psychology* 7: 268–288.

VanLehn, K. (1991). Rule acquisition events in the discovery of problem–solving strategies. *Cognitive Science, 15*(1), 1–47.

Wharton, C., and Kintsch, W. (1991). An overview of the construction–integration model: A theory of comprehension as a foundation for a new cognitive architecture. *SIGART BULLETIN* 2(4) :169–173.

# Learning Prediction of Time Series. A Theoretical and Empirical Comparison of CBR with some other Approaches.

**Gholamreza Nakhaeizadeh**
**Daimler-Benz AG, Research and Technology**
**Wilhelm-Runge Str.11, postfach 2360, 89013 Ulm, Germany**

**Abstract.** The concept of K-Nearest Neighbours (KNN) traced back to early fifties and during the last years it is investigated deeply by the statistical community. Case-Based Reasoning (CBR), which is very similar to KNN is rather new. Besides KNN and CBR one can use other statistical procedures like regression analysis or Box-Jenkins methods to perform the prediction tasks. Furthermore, it is possible to use the procedures based on neural networks and symbolic machine learning. Although learning prediction of time series is a very important task in different scientific disciplines, there is no comprehensive study in the literature which compares the performance of CBR with the performance of the other alternative approaches. The aim of this paper is to contribute to this debate from a theoretical and empirical point of view.

## 1. Introduction

Learning prediction of time series is a very important task in different scientific disciplines. In Statistics there are several, partly sophisticated, methods to perform this task. Generally, these procedures use the information available about the behaviour of the time series in the past to predict its development in the future. Box-Jenkins ARMA and ARIMA models are well-known examples for this type of procedures (Henery and Nakhaeizadeh (1993)).

Besides the information about the past values of the time series itself, one can also use other information based on the exogenous indicators which have an impact on the development of the time series. K-Nearest-Neighbours and regression analysis can be mentioned as examples for such procedures. Recently, the attention is focused also on the application of Neural Networks (Graf and Nakhaeizadeh (1993)). Some of symbolic machine learning algorithms based on ID3-concept can be used to predict the development of time series as well (Merkel and Nakhaeizadeh (1992)). It should be mentioned that although CBR, which is very similar to KNN, has found several applications for examples in classification, planning and design (see Althoff et al. (1992)), very little attention has been paid to the application of CBR to time series prediction. An exception is the work of Quinlan (1993) which applies both CBR-based and model based learning approaches to prediction.

The above facts show that several alternative approaches can be applied to prediction of time series. The aim of this study is to evaluate, firstly, these alternative approaches from a theoretical point of view and, secondly, to compare their performance in dealing with real-world prediction problems arise in industry and commerce. We will refer also to some results achieved within an Esprit-Project funded by the European Community.

## 2. A Short Description of the Applied Alternative Approaches

Before we give a summary about the theoretical aspects of different approaches which can contribute to prediction of development of time series, we should mention here a general problem exists in dealing with a large number of time series. This is the limited number of available cases. In many circumstances,

there is no information at all about the far past values of the time series. On the other hand, if such a dataset is available, it is not always suggestive to use it because too far past values have only a weak impact on the future development of the time series. It means, in dealing with time series the learning task has to be performed by using only a limited number of training data. Having this fact in mind, we will give in following a short description of different approaches.

### Linear Regression Analysis and Box-Jenkins Approach

Denoting $Y_t$ as a time series in period $t$, a linear regression model can be described by the equation

$$Y_t = a + \sum_{i=1}^{n} b_i X_{it}$$

In the above equation, $X_{it}$ denotes the value of exogenous variable $X_i$ in the period $t$. The value $Y_{t+1}$ in the period $t+1$ can be predicted simply as:

$$\hat{Y_{t+1}} = \hat{a} + \sum_{i=1}^{n} \hat{b_i} X_{i(t+1)}$$

where $\hat{a}$ and $\hat{b_i}$ are the estimations for $a$ and $b_i$ and can be calculated using least-squares or maximum-likelihood method. Of course, one can use instead of a linear regression a nonlinear model as well. In this case, the parameters $a$ and $b_i$ can be estimated using numerical procedures. The regression analysis is theoretically well investigated and it is very simple to apply. One disadvantage of this method is the problem of model selection. A lot of other statistical approaches have the same disadvantage as well. The other problem is that the calculation of $\hat{Y_{t+1}}$ is only possible when all $X_{i(t+1)}$ are known for the period $t+1$ in advance, which is in praxis not always the case.

Concerning the Box-Jenkins approach, one can describe an ARMA (autoregressive moving average) model as:

$$Y_t + \alpha_1 Y_{t-1} + .... + \alpha_p Y_{t-p} = \epsilon_t + \beta_1 \epsilon_{t-1} + .... + \beta_q \epsilon_{t-q}$$

where $\epsilon_t$ are independent normal distributed random variables.

If the parameters $\alpha$ or $\beta$ are zero, the above model will be reduced to a MA (moving average) or AR (autoregressive) process, respectively.

The main assumption in the ARMA model is that the time series $Y_t$ is stationary. A time series is stationary if its means and variance remain unchanged with the time. For a lot of real world time series, this assumption is not valid. In such cases, the time series should be transformed for example by taking successive differences so long as necessary to make the resulting series stationary. In this case, the original series is called an integrated ARMA process, i.e. an ARIMA process. Although the Box-Jenkins approach has some advantages, one needs a lot of experience to be able to apply it efficiently (see Henery and Nakhaeizadeh (1993)).

### Symbolic Machine Learning and Neural Networks

Most of the symbolic machine learning algorithms are more appropriate to perform the classification tasks. But between the ID3-type algorithms, CART and NEWID can also be used for prediction because they can handle continuous-valued classes as well. In contrast to other approaches, the predictor derived from these learning algorithms consists of a decision tree which can be transformed to production rules. Furthermore, these learning algorithms apply a single attribute at each level of the tree and this is in

contrast to the most statistical and neural learning algorithms which consider all attributes to make a decision.

The main advantage of symbolic machine learning approach is that it is possible very easily to involve other available information in prediction process, for example, by including the background knowledge of experts. However, like other approaches, prediction algorithms based on symbolic machine learning have also some shortcomings. Generally, they can not predict the values beyond the range of training data. Regarding the fact that, especially, a lot of time series have an increasing (decreasing) trend component, it can be seen that by using just the raw class values, one can never achieve a predicted value which is outside the range of the class values used for training. This can be avoided by taking differences of the class values as it was the case in Box-Jenkins approach.

In the recent years, one can also see in literature some efforts put to apply Neural Networks to prediction of time series. Although the development of Neural Networks at early stage was stimulated by modelling of learning process in human brain, the further development of this technology shows a very strong similarity with statistical approaches. There are some studies which compare the Neural Networks with some statistical procedures like nonlinear regression from a theoretical point of view (see for example Arminger (1993)). However, it should be mentioned that the ability of adaptive learning which charactrizes the most of Neural Networks is not implemented in statistical procedures like regression analysis and Box-Jenkins approach.

The main problem in using Neural Networks for prediction consists of finding the optimal network architecture. To realize this task, one has to devide the available time series data into two training and test sets. Regarding the problem of limited number of observations in time series data which is dicussed at the beginning of this section, deviding the whole series into two training and test sets leads to an still smaller training dataset, in many circumstances.

### K-Nearest Neighbours and Case-Based Reasoning

Although the concept of KNN traced back to early fifties (see for example Fix & Hodges (1951)), the studies on CBR are rather new and are mostly due to Artificial Intelligence researchers. Regarding the prediction task, KNN and CBR try to find the patterns in the past data which have the most similarity to the recent pattern $Y_{t-k}, ..., Y_{t-1}, Y_t$. The prediction value for the recent pattern is then simply the average of the prediction values of the most similar patterns in the past.

There is a controversial discussion if KNN and CBR can be regarded at all as inductive learning methods. The reason for this controversy is that the learning task in the most inductive systems generates, in contrast to CBR, a general concept which can be used later for predicting the class of unseen cases. On the other hand, it is true that in CBR one uses the information given by the cases. This information is applied, however, to measure a pre-defined distance function but it is not applied to find a general prediction concept which is the main part of inductive learning. The learning task in CBR and KNN is limited to finding similarities. Formalization of the relation between CBR and inductive concept learning is discussed by Jantke (1992).

The problems mentioned in the case of Neural Networks exist in application of CBR and KNN as well. Especially, finding the optimal length of the searched pattern and determining the number of considered patterns ($K$) need again using a separate test dataset which reduces the number of available training cases.

## 3. Empirical Evaluation Results

There are some studies in literature which compare the performance of different statistical approaches using the time series data (Makridakis et al (1984)). But, there is no comprehensive study which includes the recent developed prediction approaches based on the AI-methodology like CBR, Neural Networks and Symbolic Machine Learning. An exception is the attempts put on this task within the Esprit-Project StatLog. In this Project three real time series datasets are applied to compare the performance of different learning algorithms.

As it mentioned before, although a lot of learning algorithms can perform the classification task, they can not be applied to prediction, directly, because they can not handle the continuous-valued classes. It is, however, possible to consider the prediction task as classification by an appropriate discretization of the class values.

The first application used in the project StatLog deals with prediction of development of interest rates on successive trading days. The empirical results for this dataset are ambiguous. On one hand, some symbolic machine learnig algorithms like CN2 deliver very precise predictions. On the other hand, the performance of the other machine learning algorithms like NEWID and C4.5 are very poor. CBR-type and Neural Networks algorithms are not evaluated for this dataset. The second and the third datasets are two versions of an real-world application which is in interest of the marketing department of Mercedes-Benz AG, Stuttgart. This application deals with prediction of number of registered cars and trucks in France. While the performace of Box-Jenkins method and NEWID are the best for this application, the prediction power of a CBR-type algorithm based on the KNN-concept is very poor. Other statistical and neural networks learning algorithms deliver an average performance (see Henery and Nakhaeizadeh (1993) for more detail).

Besides the results we have achieved within the project StatLog, some other empirical works has be done by the Machine Learning Group at the Ressort Research and Technology of Daimler-Benz AG in Ulm. Besides the prediction of number of cars and trucks for the other countries, we have evaluated different learning algorithms by using another real-world application which deals with prediction of daily exchange rates of US-Dollar against D-Mark. Work on this application is in progress. The first results show that the performance of CBR, Neural Networks and Symbolic Machine Learning algorithms are almost the same. But they are still too far from the accuracy rates which one can get for example by using classical chart analysis.

## References

Althoff, K. D, Wess, S. Bartsch-Spörl, B. and Janetzko D. (Hrsg.) (1992). Proceedings of the Workshop: Ähnlichkeit von Fällen beim fallbasierten Schließen. University of Kaiserslautern. Fachbereich Informatik.

Arminger, G. (1993). Ökonometrische Schätzmethoden für Neuronale Netze. To appear in Bol, G. Nakhaeizadeh, G. and Vollmer K. H. (Eds). Proceedings of the Fourth Econometric Workshop. Physica Verlag, Heidelberg.

Fix, E. and Hodges J.L. (1951). Discriminatory Analysis, Nonparametric estimation: Consistency Properties. Report no 4, UASF Scholl of Aviation Medicine, Texas.

Graf. J. and Nakhaeizadeh, G. (1993). Application of Neural Networks and Symbolic Machine Learning to Predicting Stock Prices. To appear in : Plantamura, V. L. Soucek, B. and Visaggio, G. (Eds). Logistic

and Learning for Quality Software, Management and Manufacturing. Wiley & Sons , New York.
Henery, R. and Nakhaeizadeh, G. (1993). Forecasting of Time Series. Mimeo, University of Strathclyde, Glasgow.
Jantke, K. P. (1992). Formalizations in Case-Based Reasoning. In : Althoff, K. D, Wess, S. Bartsch-Spörl, B. and Janetzko D. (Hrsg.) (1992). Proceedings of the Workshop: Ähnlichkeit von Fällen beim fallbasierten Schließen. University of Kaiserslautern. Fachbereich Informatik, 9-14.
Makridakis, S. Andersen, A. Carbone, R. Fildes, R. Hibon, M. Lewandowski, R. Newton, J. Parzen, E. and Winkler, R. (1984). The Accuracy of Extrapolation ( Time Series) Methods: Results of a forecasting competition. In : Makridakis, S. (Ed.). The Forecasting Acuuracy of Major Time Series Methods. Wiley & Sons. 103-166.
Merkel, A. and Nakhaeizadeh, G. (1992). Application of Artificial Intelligence Methods to Prediction of Financial Time Series. In: Gritzmann, p. et al. ( Hrsg.). Operations Research 91, 557-559.
Quinlan, J. R. (1993). Combining instance-based and model-based learning. Proceedings of the Tenth International Conference on Machine Learning, 236-243. Morgan Kaufmann Publishers.

# Incorporating (Re)-Interpretation in Case-Based Reasoning

Scott O'Hara and Bipin Indurkhya

College of Computer Science
Northeastern University
Boston, MA. 02115, USA
Email: {bipin|ohara}@ccs.neu.edu
Tel: 617-373-5204; Fax: 617-373-5121

## 1 Introduction

One advantage of case-based reasoning over rule-based reasoning that has been advocated is that cases can be interpreted differently, whereas once a rule has been abduced from cases, there is no possibility of reinterpreting the cases. For instance, Riesbeck and Schank (1989, pp. 9-14) compare and contrast three modes of reasoning: 1) reasoning with ossified cases (rules or abstract principles), 2) reasoning with paradigmatic cases (cases with a given interpretation), and 3) reasoning with stories (cases with many possible interpretations and capable of re-interpretation). They argue that it is the third mode of reasoning that displays the most flexibility and power of having a knowledge base containing cases.

However, most of the existing work on case-based reasoning remains confined to the second mode or to a version of the third mode where cases have a number of fixed interpretations. Almost all existing case-based reasoning systems associate dimensions (also called indices) with every case in a case-base, and use these dimensions for similarity assessment and retrieval. Consider, for instance, the system Hypo that applies case-based reasoning to law (Ashley 1990). At the time each case is entered in the case base, one must determine the possible ways in which that case might be relevant, and each relevant factor that is found is assigned a dimension. As an example, take the domain of home-office tax deduction that is discussed in Rissland and Skalag (1991). Upto a point in time, the courts were consistently ruling that the statutory predicate 'principal place of business' means the place where the most important part of the business is carried out, which would mean classroom for a teacher, concert stage for a musician etc. However, in one particular case the court decided that this was an unfair test, and decided to consider the place where the taxpayer spends the most amount of their time as the principal place of business, which could be home-office for a teacher, if she spends most of her time there in preparing for lectures, grading, etc. This decision, however, introduces a fresh dimension, for now we must consider the place where the taxpayer spends most time in arguing a case, and citing a precedent. If this dimension was not included in the cases that are already in the case base, the retrieval mechanism will miss out on many relevant precedents.

Of course, one solution is to include a large number of dimensions with each case. But then this will have the disadvantage that many irrelevant cases will be retrieved, not to mention the fact that there is always the possibility that, no matter how large the initial set of dimensions, a new dimension becomes necessary that was not foreseen before and hence not included in the initial set.

This point is best illustrated with the domain of geometric figures. Consider Fig. 1. Suppose this

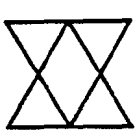


Figure 1: A figure included in a case base.

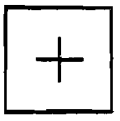


Figure 2: Another figure to be interpreted in terms of Fig. 1.

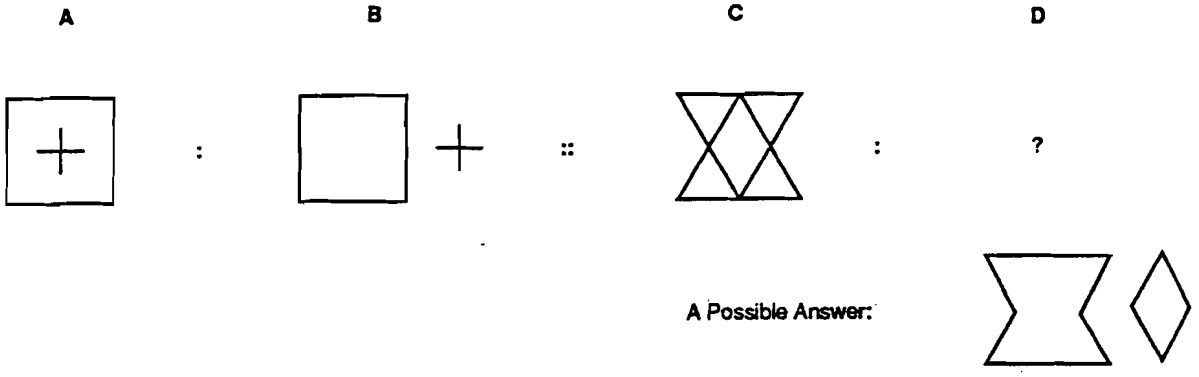



Figure 3: A proportional analogy including Fig. 1.

figure is to be included in a case base. How should we dimension it? Perhaps one obvious way is to dimension it is as a figure consisting of four triangles. But then Fig. 2 would not be seen as similar to Fig. 1 at all, whereas, given the right context (Fig. 3) the similarity between the two becomes obvious. Moreover, no matter how many dimensions were used in the initial representation of Fig. 1, we can always produce another figure that is similar to it, but requires a new dimension.

It is clear that what is necessary is a way to interpret the figure differently depending on the context, and *create* new dimensions or indices as appropriate. This is the ultimate promise of case-based reasoning, as rightly emphasized by Riesbeck and Schank, that is yet to be delivered.

We have been working towards fulfilling the promise of case-based reasoning. One of the authors (Indurkhya 1991, 1992) has been working on formalizing the process of reinterpretation in an algebraic framework, and on articulating the crucial role it plays in many aspects of cognition. The other author (O'Hara 1992) has been implementing a system PAN that models this reinterpretation process in the domain of geometric figures. PAN is designed to solve proportional analogy relations of geometric figures that involve reinterpretation. We will present an outline of the architecture of PAN in Section 2. In Section 3 we discuss briefly how the reinterpretation mechanism *a la* PAN can be incorporated in a conventional case-based reasoning system. In Section 4, we point out the further research questions that are raised by our approach.

## 2 The Architecture of PAN

PAN (for Proportional ANalogy) is a program being developed to solve geometric proportional analogy problems. The input to PAN consists of three geometric figures A, B and C made up of straight line segments. The output of PAN is a new geometric figure D such that the four figures, A, B, C and D satisfy the proportional analogy relation: A is to B as C is to D. PAN creates the answer figure D by constructing descriptions of the figures A, B and C. These descriptions are at a higher "conceptual-level" than the initial line-segment input and involve rotations, translations, repetitions, convex polygons, symmetry etc. A key feature of PAN is that the descriptions of the figures are constructed "in tandem" permitting the figures and their descriptions to act as contexts for each other during their construction.

The architecture of PAN is illustrated by the diagram in Fig. 4 and is essentially that of a production system. In the diagram, circular and oval shapes represent data structures and rectangular shapes represent processes. The input to PAN is represented by the oval containing the geometric figures in the lower-left part of the diagram. The input is simply a set of line segments representing the figures A, B and C. The first task performed by PAN is to preprocess the raw input data into a graph-like structure that makes computations easier and allows PAN to keep track of what part of the figure has been described and what part remains to be described. One such graph is constructed for each of figures A, B and C.

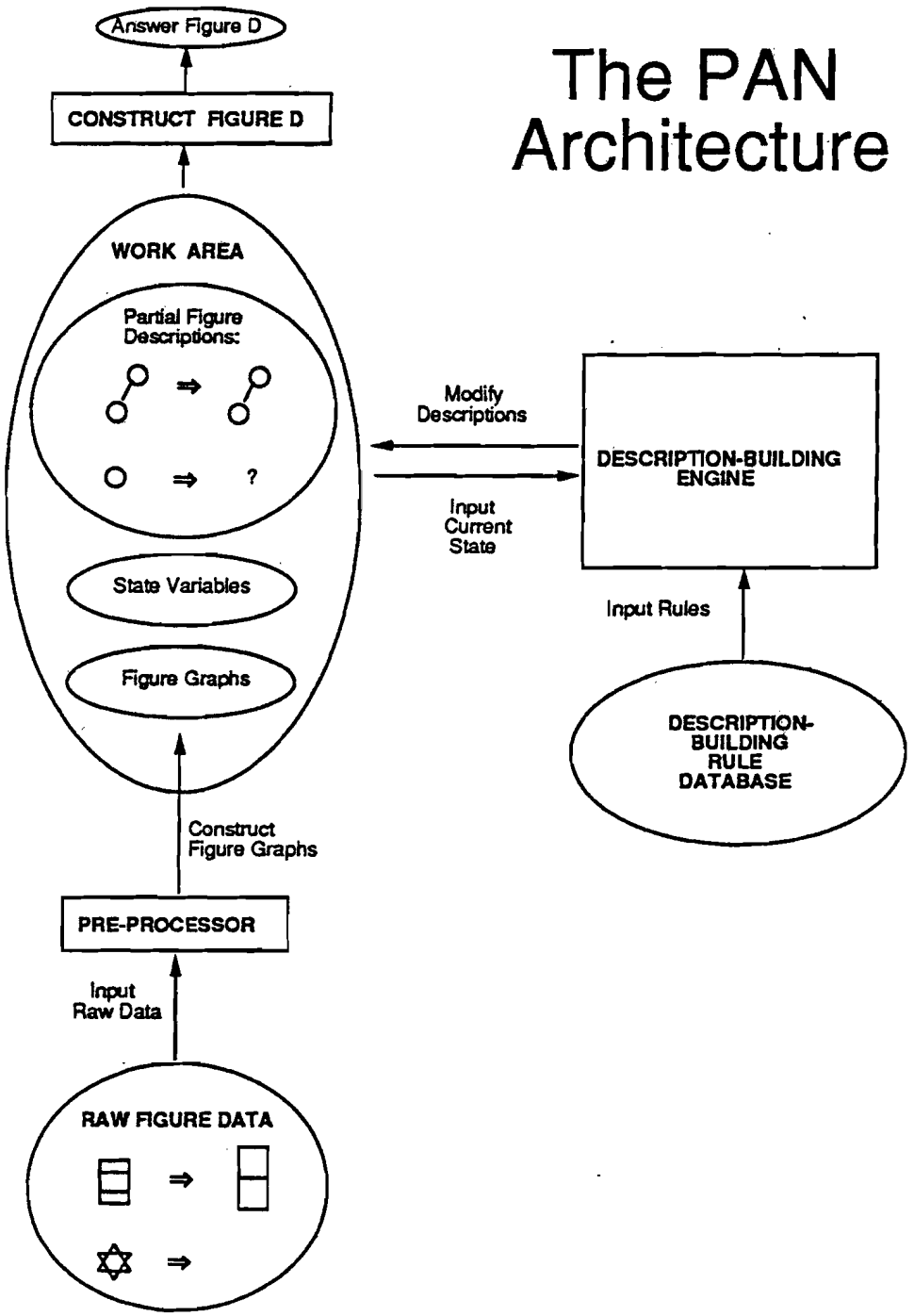



Figure 4: The Architecture of PAN.

After PAN finishes the preprocessing step, it enters into a search process that builds the descriptions of the figures. This process is represented by the block called *the description-building engine.* The description-building engine takes as input the contents of the *work area* and a data base of condition-action rules called the *description-building rule database* (or rule database for short). The work area consists of partial descriptions of figures A, B and C (initially, the descriptions are null), the figure graphs constructed by the preprocessor, and a vector of state variables which are tested in the condition part of the rules and used to indicate what task should be done next in the work area.

The description-building engine checks the conditions of the rules against the contents of the work area and then modifies the contents of the work area by applying the actions of a matched rule. Typically, the actions of a rule will do three things: (1) extend a description in the work area, (2) update the corresponding figure graph to reflect that more of the figure has been described, and (3) modify the state variables to indicate what should be done next. This process continues until all figures are completely described, at which point the fourth figure D is generated. In general, the description-building engine will find more than one rule that applies. These options are tried one by one, backtracking whenever an option fails or proves to be too complicated.

The overall search process executed by the description-building engine is guided by an iterative-deepening search strategy (Korf 1985). Iterative-deepening visits the nodes of a search tree by executing a series of bounded depth-first searches to an ever increasing depth in the tree. The depth in the tree to which the depth-first search goes is specified by a depth variable. This variable is first set to one so the first depth-first search just visits the root node. If the goal is not found, the depth variable is incremented so that the next depth-first search visits the root node and all the children of the root. The depth variable is repeatedly incremented and depth-first searches are repeatedly performed until a goal node is obtained. While this strategy appears to be wasteful since it covers the same nodes over and over again, it actually performs much better than a breadth-first search and is guaranteed to find a shortest path to a goal node.

In the description-building engine, the depth variable represents the overall complexity of the descriptions in the work area which is measured by the *description complexity function.* This function is defined on the positive integers and can be arbitrarily large. At present, the description complexity function is a count of the number of different description elements that appear in the descriptions in the work area. Combinatorial explosion is delayed by placing a limit on the size of each individual description and by limiting the number of "combinatorial" description elements that may appear in any one description. This approach of preferring the least complex descriptions is similar to the approach of van der Helm, van Lier and Leeuwenberg (1992) who deal with the description of individual geometric figures. While van der Helm et al. have focused on the problem of finding the least complex description for single figures, our approach focuses on finding the least-complex overall description of a proportional analogy. In our framework, it is possible that the description of an individual geometric figure will be different depending on the proportional analogy in which it occurs.

The description language in which the geometric figures are represented is an algebra-like construction consisting of a set of primitive objects (polygons and broken line-segments) and a set of operations which transform geometric objects into new ones. There are three broad classes of operators that we use: (1) *iterative processes* which make multiple copies of a figure. For example, in Fig. 1, the top two triangles might be obtained by applying an iterative process to the upper-left triangle. The entire figure can be obtained by applying another iterative process to the top two triangles obtaining the bottom two triangles; (2) *join operators* are multiple-argument operators which combine two or more geometric figures into a single composite figure. A join operator will typically require that argument figures have some particular relationship to one another and may require that an argument be of a particular type such as a polygon. For example, Fig. 2 is constructed by applying an 'inside' join operator to a square and the cross figure; (3) *global operators* are single argument operators that act on a figure as a whole. Examples of these operators are: rotate, scale, stretch etc. These three classes of operators were arrived at empirically by examining a number of typical proportional analogies.

A description of a geometric figure is modeled as a *description tree* with exterior nodes labeled with objects and interior nodes labeled with operators. The object described by a description tree is found by recursively evaluating the tree. A description tree is essentially a possible history of how the geometric figure might have been constructed. When forming a proportional analogy, descriptions of figures A and B are related to one another by substituting, inserting and deleting operators in description A to form description B. Descriptions A and C must be isomorphic.

The production rules in PAN come in several varieties. We illustrate these rules by means of two examples. Consider how PAN might solve the proportional analogy in Fig. 3. The first thing that PAN might do is recognize the square in figure A. Next, a new rule would detect that the square in figure A

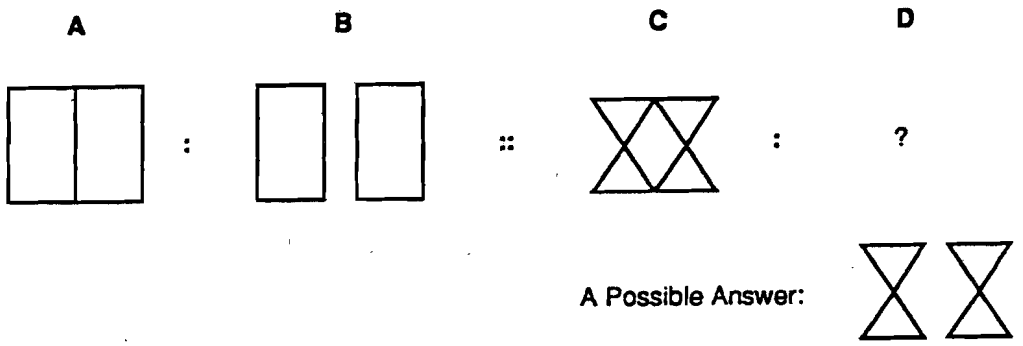



Figure 5: Another proportional analogy involving Fig .1.

is similar to the square in figure B. Figures A and B are then partially described as containing squares. Next, PAN might notice that the cross in figure A is similar to the cross in figure B. The descriptions of A and B are then expanded to reflect this. Given that there are two objects noticed so far in A, a rule may be applied to determine what their relationship is. In this case, a 'inside' join operator would be inserted into A's description. Similarly, a 'left-of' join operator would be inserted into B's description. Notice that both A and B are fully described, so there is nothing else to do but for the description of A to interact with figure C. Given that figure A is currently described as a cross inside a square, PAN might then attempt to describe figure C similarly. PAN, using a *projection procedure* attached to the 'inside' concept, decomposes C into a "containing" polygon and an inner figure (the diamond.) From these full desciptions of A, B and C, the figure D is generated.

Figure. 5 shows a proportional analogy that would result in a different description of figure C. After preprocessing figures A, B and C, the first thing that PAN might do in this example is to find a the left-hand rectangle in A. Next, a new rule would detect that the left-hand rectangle in A is similar to the left-hand rectangle in B. An iterative process now might be inserted above the left-hand rectangle in A providing a description of the whole figure A. Similarly, a different iterative process might be inserted above the left-hand rectangle in B providing a description of the the whole figure B. Figures A and B are now fully described, so the description of A must now interact with figure C. PAN uses a the projection procedure attached to the iterative process in A to decompose C. (The behavior of this procedure is rather involved, so we will leave it to the full paper to describe.) C is now described as two hour-glass figures that have been pulled apart.

# 3 Interpretation in Case-Based Reasoning

In the introduction, we articulated the need for a reinterpretation component in case-based reasoning. We do not propose it as an alternative to the conventional approach using dimensions (or indices) but *in addition to it.* It should be clear from our brief description of the PAN architecture in the last section that it is a computationally expensive process. Moreover, when an aspect of a case is deemed relevant, and turned into a dimension, it is usually because it is considered to have more general appeal than just as an idiosyncracy of that case. Therefore it seems quite likely that many new problems could be solved using conventional dimensions, which allow a fast retrieval of similar past cases.

So it would be prudent to continue to encode the cases in terms of dimensions depending on what aspects of it seem relevant at the time the case is entered in the case base. But then we could provide an interpretation module that is evoked when the retrieval based on conventional dimensions is not helpful. This could be because the retrieved cases, even though they are similar to the problem, do not have solutions that can be easily adapted to solve the problem (Börner 1993), or it could be because the problem at hand requires attention to an aspect that was not considered relevant so far, and is therefore not included in the dimensions. In all such situations, the reinterpretation mechanism is called, which alters the similarity metric (as manifested by existing dimensions) so that the cases in the case base are made to look similar to the new problem, like making Fig. 2 seem similar to Fig. 1.

It may seem at first that the reinterpretation process is rather like a runaway horse, retrieving a horde of useless cases from the case base, for almost anything could be made to look similar to anything else. However, a careful analysis in any domain shows that there are sufficient top-down, goal-directed constraints to keep the reins on reinterpretation. For instance, in the domain of geometric proportional

analogy, the context provided by the other figures acts as a powerful constraint to focus the search for new dimensions in the right direction. In the domain of legal reasoning, which one of the authors has been exploring and where there is a crucial need for reinterpretation mechanism, we have found that the goals of the arguer serve as a beacon to keep the search for new dimensions and relevant precedents from growing exponentially (Jantezko and Indurkhya, in preparation).

# 4 Conclusions and Further Research

We have argued in this paper for a need to incorporate a reinterpretation mechanism in case-based reasoning systems, and have outlined an approach to it. Obviously, we are just crossing the threshold into a new realm where a lot of exploration needs to take place. Our work on modeling reinterpretation in geometric proportional analogies and legal reasoning is only a beginning of this exploration. We hope, however, that other researchers working on case-based reasoning would also realize the need to address the process of reinterpretation and join this exploration. Only then we will be able to realize the full potential of case-based reasoning.

**Acknowledgements.** The work described in this paper was supported by National Science Foundation grant IRI-9105806.

# 5 References

Ashley K.D., 1990, *Modeling Legal Argument: Reasoning with Cases and Hypotheticals,* MIT Press, Cambridge, Mass.

Börner K., 1993, "Structural-Similarity as Guidance in Case-Based Design," submitted to European Workshop on Case-Based Reasoning.

Indurkhya B., 1991, "On the Role of Interpretive Analogy in Learning," *New Generation Computing 8,* No. 4, pp. 385–402.

Indurkhya B., 1992, *Metaphor and Cognition,* Kluwer Academic Publishers, Dordrecht, The Netherlands.

Janetzko D. and Indurkhya B., in preparation, "Toward a Model of Reinterpretation in Legal Reasoning."

Korf, R. E., 1985, "Depth-first iterative deepening: An optimal admissible tree search," *Artificial Intelligence,* 27(1):97-109.

van der Helm, P. A., van Lier, R. J., and Leeuwenberg, E. L. J., 1992, "Serial Pattern Complexity: Irregularity and Hierarchy," *Perception,* Volume 21, pp. 517-544.

O'Hara S., 1992, "A Model of the 'Redescription' Process in the Context of Geometric Proportional Analogy Problems;" in K.P. Jantke (ed.) *Analogical and Inductive Inference,* Lecture Notes in Artificial Intelligence 642, Springer-Verlag, Berlin, Germany, pp. 268–293.

Riesbeck C.K. and Schank R.C., 1989, *Inside Case-Based Reasoning,* Lawrence Erlbaum and Associates, Hillsdale, New Jersey.

Rissland E.L. and Skalag D.B., 1991, "Cabaret: Rule Interpretation in a Hybrid Architecture," *International Journal of Man-Machine Studies 34,* pp. 839–887.

# PBL: Prototype-Based Learning Algorithm

Kuniaki Uehara, Masayuki Tanizawa and Sadao Maekawa
uehara@jedi.seg.kobe-u.ac.jp
Department of Computer Science and Systems Engineering
Kobe University
Nada, Kobe 657, Japan

**Abstract**

In this paper, we will introduce an inductive learning algorithm called Prototype-Based Learning (PBL). PBL learns a concept description, which consists of both prototypical attributes and attribute importances, by using a distance metric based on prototype-theory and information-theory. PBL can learn the concept description from even a small set of training cases and is tolerant of inappropriate cases. Furthermore, even the attribute importance differs depending on the combinations of the other attribute-value pairs present describing the case, PBL can learn the concept description and highly utilize it so as to do the accurate classification. Finally, PBL can learn indexing knowledge directly from the concept description, which is useful for a human expert to understand and verify the concept description generated by the learning algorithm.

## 1 Introduction

This paper describes an overview of the approach we are taking to machine learning within a continuing research project. The project is concerned with developing a cognitively based symbolic concept learning algorithm. In contrast, the type of machine learning that has attracted most attention in the AI literature is the learning of minimum discrimination rules to classify a new case into an appropriate category. In this type of machine learning, if a training set involves inappropriate cases (i.e. noisy cases, incomplete cases, or exception cases), the approach of extracting discrimination rules may generate too complex and failure rules. On the other hand, our approach is to classify a new case into nearest categories by use of a distance metric based on prototype-theory [6] and information-theory. More precisely, our approach extracts a concept description, which consists of both prototypical attributes and attribute importances, from the set of training cases. As a result, our approach can learn the concept description from even a small set of training cases and is tolerant of inappropriate cases. Furthermore, even the attribute importance is context-sensitive, our approach can extract and highly utilize the concept description so as to do the accurate classification. Finally, our approach can learn indexing knowledge directly from the concept description, which is useful for a human expert to understand and verify the concept description generated by the learning algorithm.

## 2 Basic Ideas

### 2.1 Prototypicality Ratings

PBL1 is the simplest prototype-based learning algorithm. The learning task of PBL1 is inductive learning or learning by examples. Especially, we will focus on probabilistic approach to inductive learning in which the only input is a sequence of cases. Each case is assumed to be represented by a set of attribute-value pairs and its category. For example, if the $j$-th case $I_j$ has the $n$ attributes $a_1$, $a_2$, ... , $a_n$ and the category is $c$, the case $I_j$ is represented as follows:

$$I_j = (c, a_{1j}, a_{2j}, \ldots, a_{nj})$$

where $j$ ranges over the cases in the training set.

The primary output of PBL1 is a concept description. This is a function that maps cases to categories. The concept description is represented by a distance metric called prototypicality ratings. Prototypicality ratings provide a partial ordering on candidate categories. That is, prototypicality is a rating of the representativeness of a case with respect to a category. Thus, the category which has the highest family resemblance is the most prototypical. During classification, PBL1 uses prototypicality ratings to determine the category which is the most likely to match the new case. To compute prototypicality ratings,

PBL1 counts the frequency of each value that the attribute can take on. If the new case $I_{new}$ is given, the prototypicality ratings can be computed as follows:

$$prototypicality(c, I_{new}) = \sum_{i=1}^{n} \frac{\sum_{j=1}^{N} f(a_i, a_{ij})}{N}$$

$$\text{where } f(a_i, a_{ij}) = \begin{cases} 1 & \text{if } a_i = a_{ij} \\ 0 & \text{otherwise} \end{cases}$$

where $i$ ranges over the attributes. $a_i$ is the value of the case $I_{new}$ on attribute $i$. $j$ ranges over the cases in the category $c$. $N$ is the total number of cases in $c$.

## 2.2 Empirical Studies with PBL1

Fig. 1 presents the experimental evidence for the performance of PBL1. In this experiment, the performance accuracy of PBL1 was compared to ID3 [5] on the "famous" soybean database which contains the diagnosis of soybean diseases. The database contains 289 cases and 17 categories (diagnoses). Diagnoses are described by 50 attributes (plant and environmental descriptors). We used 145 cases as the training set and 144 cases as the test set. The training and test sets were always disjoint. These cases were drawn randomly from the database. All results reported in Fig. 1 were averaged over 20 trials.

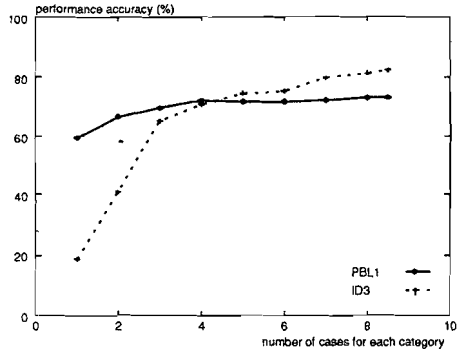



Figure 1: Performance accuracy of PBL1 and ID3.

Fig. 1 shows that PBL1 behaves well, but the performance accuracy of ID3 becomes greater than that of PBL1 as the number of cases for each category increases. Furthermore, performance accuracy of PBL1 does not increase even if the number of cases increases, whereas performance accuracy of ID3 increases as the number of cases increases. From this experimental result, we will not argue that PBL1 is superior to the more sophisticated algorithm ID3, but the result shows that it behaves well even the number of cases for each category is small. Thus, PBL1 is a promising learning algorithm that deserves more intensive extension.

# 3 Some Extensions to Prototype-Based Learning

## 3.1 Attribute Importance

PBL1 differs from ID3 in the following important respect. PBL1 learns salient (prototypical) attributes while ID3 learns discriminant attributes. Salient attributes represent what cases in a single category have in common. Discriminant attributes represent distinguishing attributes of two or more categories. When PBL1 computes the prototypicality ratings, it is learning the salient attributes of a category. However, PBL1 assigns the same weight setting to each attribute, and does not pay attention to the relative importances of attributes. Thus, PBL1 performs poorly if the training set involves large numbers of irrelevant attributes. This leads to the development of PBL2, which learns the relative importances of attributes, represented as attribute weight settings, for the purpose of computing accurate similarity assessments.

Now we will introduce an information-theoretic approach and augment the prototypicality ratings described in Section 2. PBL2 examines all attributes and computes the expected information for each attribute. Intuitive basis for this approach is that an attribute distributed in many categories has high expected information, whereas an attribute occurring in only one or two categories has lower expected information. Therefore we use the expected information as a measure of attribute importance. However,

since an unimportant attribute for the purpose of accurate classification, that is, an attribute distributed in many categories, should be ignored, the attribute weight setting should have a low value, whereas the important attribute should be assigned the greater attribute weight setting. Thus, the attribute weight setting is defined as follows:

$$\omega(a_i) = 2^{-\left(-\sum_{c_l \in C} p(c_l|a_i) \log_2 p(c_l|a_i)\right)} = \prod p(c_l|a_i)^{p(c_l|a_i)}$$

where $l$ ranges over the categories. $p(c_l|a_i)$ means the probability that the observed case with attribute $a_i$ will be determined to belong to the category $c_l$ and $\sum p(c_l|a_i) = 1$. Furthermore, the range of $\omega(a_i)$ is $[0, 1]$: $\omega(a_i) = 1$ means the maximum attribute weight setting.

Now that we have defined the attribute weight settings, we will augment the definition of prototypicality ratings introduced in Section 2.1. The definition of augmented prototypicality rating is defined as follows:

$$prototypicality(c, I_{new}) = \sum_{i=1}^{n} \frac{\sum_{j=1}^{N} f(a_i, a_{ij})}{N} \times \omega(a_i)$$

$$\text{where } f(a_i, a_{ij}) = \begin{cases} 1 & \text{if } a_i = a_{ij} \\ 0 & \text{otherwise} \end{cases}$$

Fig. 2 shows that PBL2 outperforms PBL1 and ID3. To some extent, the experiment shown in Fig. 2 also indicates that PBL2 can tolerate irrelevant attributes better than PBL1, which effectively assigns the same (static) attribute weight setting to each attribute. This is because PBL2 learns a separate set of attribute weight settings for each category. Since attribute weight setting is used as the similarity function, learning attribute weight setting is in effect learning a separate similarity function for each category. Therefore, even if the given case includes some irrelevant attributes, their attribute weight settings are relatively low and do not affect the similarity assessments.

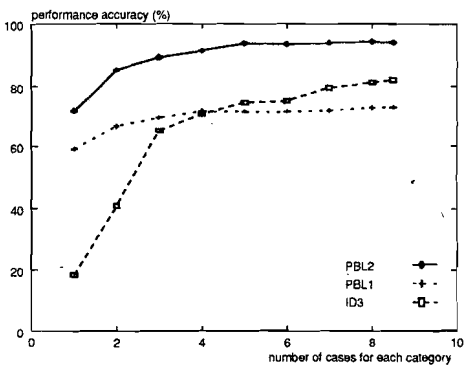



Figure 2: Performance accuracy of PBL1, PBL2 and ID3.

## 3.2 Attribute Importance in Context

PBL2 will not perform particularly well where attribute importance is context sensitive, in the sense that the attribute importance differs depending on the candidate categories under consideration. Consider the following example, a taxonomy of vehicles:

| | |
|---|---|
| superordinate: | vehicles |
| subordinate: | bicycle, moped, motorcycle, passenger cars, van, pickup, ... |

In this situation, the attribute 'two-wheels' would be expected to be important, if we discriminate between 'motorcycle' and 'passenger car.' However, this attribute should be assigned lower importance, if we discriminate between 'motorcycle' and 'bicycle.' That is, we cannot assign a single attribute weight to each attribute ahead of time; instead, the weight must be re-calculated with respect to the candidate categories under consideration.

Context sensitive attribute weight settings are required to derive appropriate attribute importance in applications where attribute importance is context-dependent. Since PBL2 adopts one-shot approach

which considers all training cases and computes both prototypicality ratings and attribute weight settings at one time, PBL2 cannot deal with the context sensitive problem.

Now we will propose the two-step approach to prototype-based learning algorithm named PBL3. The first stage of PBL3 is quite similar to PBL2 in that it considers all training cases at one time, but it also provides a partial ordering on categories based on prototypicality ratings. In the second stage, PBL3 selects the most promising categories (i.e. the category whose prototypicality rating is the highest and its nearest neighbors) and recomputes the prototypicality ratings among them. Prototypicality ratings in the second stage of PBL3 is modified so as to amplify the attribute importance. The modified definition is as follows:

$$prototypicality(c, I_{new}) = \sum_{i=1}^{n} \frac{\sum_{j=1}^{N} f(a_i, a_{ij})}{N} \times \omega(a_i)^x$$

where the range of the amplifier $x$ is $[1, 4]$. In other words, the first stage of the classification process is to learn a prototype for each category. The second stage is to use these prototypes so as to discriminate among the similar candidate categories. Fig. 3 shows the performance accuracy of PBL3 and its ancestor PBL2. The results of the comparison shows that PBL3 outperforms PBL2 and achieves a high classification accuracy as the number of cases for each category increases.

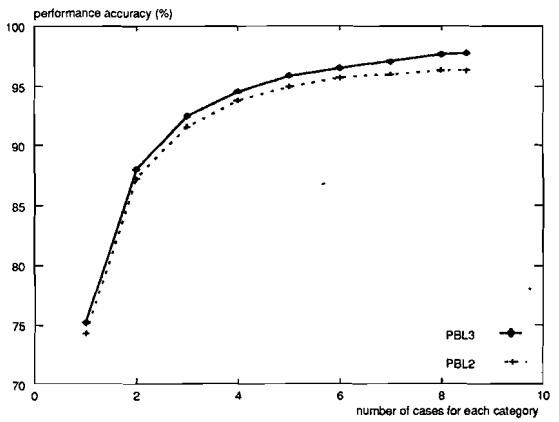



Figure 3: Performance accuracy of PBL2 and PBL3.

## 3.3 Some Experimental Results

The performance accuracies of PBL2 and PBL3 were compared to PROTO-TO [2], C4 (ID3), and Bayesian classifier [1] on eight databases. All databases except soybean data were taken from the UCI database collection [3]. The results of the comparison are shown in Table 1. Each experiment was repeated 50 times so the numbers are averages. The experimental results of PROTO-TO and C4 (ID3) were borrowed from [2].

Table 1: Comparison of performance accuracy.

| Name | PBL3 | PBL2 | PROTO-TO | C4 (ID3) | Bayesian classifier |
|---|---|---|---|---|---|
| glass | 45 - 55 % | 43 - 50 % | 48.0 % | 65.5 % | n/a |
| hepatitis | 84.2 % | 84.2 % | 79.9 % | 79.8 % | 84.8 % |
| house-vote | 92.1 % | 92.1 % | 90.4 % | 95.3 % | 90.5 % |
| soy bean data | 97.1 % | 96.3 % | n/a | (82.3) % | 91.5 % |
| breast cancer | 95.2 % | 95.2 % | n/a | n/a | 97.2 % |
| iris | 95.4 % | 95.4 % | 96.0 % | 94.2 % | 95.3 % |
| zoo | 92.8 % | 93.0 % | n/a | n/a | 93.8 % |
| tic-tac-toe | 72.9 % | 72.9 % | n/a | n/a | 67.7 % |

Table 1 shows that PBL3 recorded higher accuracies than the others in the five domains. However, in the two of the domains, such as 'hepatitis' and 'tic-tac-toe,' PBL3 performed slightly poorly, although PBL3 is at least as accurate as Bayesian classifier. In the 'glass' domain, PBL3 performed relatively poorly in comparison to the other algorithms, since the 'glass' database consists of continuous attributes instead of nominal attributes. Note that all the continuous attributes were pre-processed by using a clustering algorithm (i.e. k-means method) to ensure that they are treated with equal importance by the prototypicality ratings of both PBL2 and PBL3. Furthermore, since some databases contain only two

categories (i.e., hepatitis, house-vote, breast cancer, tic-tac-toe), the performance accuracies of PBL3 are same as those of PBL2 on these databases.

# 4 Learning Indexing Knowledge

The last problem we have to consider is to learn indexing knowledge from cases. In a domain-specific system that uses case-based learning algorithm, the efficient use of cases during classification requires that they should be indexed so that they can be efficiently retrieved when they are likely to be similar to a new case. One of PBLs' primary learning tasks is the acquisition of indexing knowledge from cases.

Three types of indexing knowledge can be obtained directly from the concept description generated by PBLs: confirmatory index, attribute-to-category index, and category-to-attribute index. These indices can be extracted from both the prototypicality ratings and attribute weight settings in the following ways:

1. If the prototypicality is equal to 1 and the attribute weight setting is also equal to 1, then extract the category $c_j$ and the attribute $a_i$. These category-attribute pairs are called confirmatory indices. Confirmatory indices associate attributes with categories and they are foolproof. That is, confirmatory index from $a_i$ to $c_j$ suggests that $c_j$ is the reliable classification for cases described with $a_i$ and that every case in $c_j$ has the attribute $a_i$ without uncertainty.

2. If the prototypicality rating is less than 1 and the attribute weight setting is equal to 1, then the attribute-to-category index is extracted. Attribute-to-category index also associates the attribute $a_i$ with the category $c_j$, but it is not foolproof. That is, the attribute-to-category index simply enumerates a set of possible classifications for the new case described with $a_i$.

3. If the prototypicality is equal to 1 and the attribute weight setting is less than 1, then the category-to-attribute index is extracted. Category-to-attribute index associates a category with an attributes. This is opposite to the direction of an attribute-to-category index. In other words, category-to-attribute indices can produce what may be called "prototypical" cases by creating a case that has the most frequent value for each attribute.

The following are examples of indexing knowledge for soybean diseases. The right hand side of the index is a triple of the form:

[attribute=value, prototypicality, attribute weight]

- confirmatory index:

  powdery mildew $\leftrightarrow$ [leaf mildew growth = on upper leaf surface, 1.0, 1.0].

- attribute-to-category index:

  phytophthora $\leftarrow$ [external decay of stem = watery and soft, 0.88, 1.0].
  phytophthora $\leftarrow$ [external stem discoloration = dark brown, 0.88, 1.0].

- category-to-attribute index:

  phyllosticta $\rightarrow$ [leaf spot color = tan, 1.0, 0.50].
  phyllosticta $\rightarrow$ [leaf spot growth = from edge of leaf inward, 1.0, 0.65].

By index transformation, we extracted 2 confirmatory indices, 27 attribute-to-category indices, and 30 category-to-attribute indices from the soybean database. Note that, in the category-to-attribute index, the number of indices depends on the threshold of the attribute weight setting. In the above example, the threshold is set to 0.3, that is, $0.3 \leq \omega(a_i) < 1.0$. If we decrease the threshold from 0.3 to 0.2, we can extract 125 category-to-attribute indices, although some of them are 'weak' category-to-attribute indices. However, an experiment was conducted in the domain of soybean disease to determine the effect of providing 'weak' category-to-attribute indices. Table 2 shows that category-to-attribute indices provided important evidence during the classification process. Consequently, removing 'weak' category-to-attribute indices actually hurt the performance accuracy of the system.

Learning indexing knowledge from cases has already been proposed by Protos [4]. Protos elicits and refines domain knowledge by interacting with a human expert in the context of problem-solving failures: failures to classify cases and failures to explain its classification. Although Protos' knowledge acquisition process is rather systematic, such a process is still very time and effort consuming for the human expert.

Table 2: Contribution of category-to-attribute indices to performance accuracy.

| | threshold value of $\omega(a_i, c_j)$ | | | | | | |
|---|---|---|---|---|---|---|---|
| | 0.1 | 0.2 | 0.3 | 0.4 | 0.5 | 0.6 | 0.7 |
| number of indices | 437 | 125 | 30 | 13 | 8 | 4 | 0 |
| performance | 87 % | 83 % | 84 % | 79 % | 76 % | 77 % | 70 % |

Furthermore, when Protos assigns the strength to each index, Protos relies heavily on the heuristic processing of explanations, which are based on a large number of underlying assumptions. This kind of approarch may be failed, if the appropriate knowledge for understanding the explanation could not be made generally available to the system *a priori*.

# 5 Concluding Remarks

In this article, we described one of the inductive learning paradigm called PBL. The PBL paradigm supports relatively robust learning algorithms. They can tolerate noisy and irrelevant attributes and can represent both probabilistic and symbolic concept descriptions. The PBL paradigm is a promising approach and is rich with opportunities for additional research.

Firstly, we have not yet studied how the PBL approach can use continuously-valued attributes to classify a case which consists of the values of either unordered or totally-ordered attributes. We are now developing a labeling procedure for the continuous attributes. The labeling procedure divides all numerical data into several clusters, and labels a new nominal attribute for each cluster. The labeling procedure will be unified into the PBL algorithm so that each numerical attribute is directly translated into the nominal attribute.

Secondly, the PBL paradigm lacks the mechanism to deal with information about which combinations of attributes comprise realizable cases of a category. The PBL paradigm also cannot represent knowledge of correlated groups of attributes nor knowledge about the acceptable ranges of values for individual attributes. This limitation confronting the PBL paradigm is problematic, and constructive induction approach may be useful to solve the problem.

Thirdly, one of the limitations of the PBL paradigm comes from the attribute-value representation for cases. The PBL algorithms cannot learn in knowledge-rich domains that require more elaborate and complex case representations. Applications involving higher-order attribute relationships, such as planning and reasoning, are not amenable to current PBL algorithms.

Finally, additional study of the PBL paradigm in the context of a large-scale database is necessary. PBL algorithms perform well in a small domain, but its storage requirements increases, since we must store the prototypicality rating and attribute weight setting for each attribute-value pair, although their required space is rather sparse. For example, 3,451 prototypicality ratings must be stored for soybean database, although about 2,000 ratings are equal to 0. In order to reduce storage requirements, we can make use of indexing knowledge extracted from the concept description. Protos also proposed the method for learning indexing knowledge from classification and discrimination failures. We cannot adopt Protos' approach directly, since it learns the indexing knowledge by being told from the human expert. However, learning the indexing knowledge from failures is an interesting topic for future research.

# References


[1] Langley, P., Iba, W. and Thompson, K.: An Analysis of Bayesian Classifiers, *Proc. of the Tenth National Conference on Artificial Intelligence*, pp.223–228 (1992).

[2] Maza, M.: A Prototype Based Symbolic Concept Learning System, *Proc. of the Eighth International Workshop on Machine Learning*, pp.41–45 (1991).

[3] Murphy, P. M. and Aha, D. W.: *UCI Repository of machine learning databases* [Machine-readable data repository]. Irvine, CA: University of California, Department of Information and Computer Science (1992).

[4] Porter, B. W., Bareiss, R. and Holte, R. C.: Knowledge Acquisition and Heuristic Classification in Weak-Theory Domains, *Artificial Intelligence*, Vol.45, pp.229-263 (1990).

[5] Quinlan, J. R.: Induction of Decision Trees, *Machine Learning*, Vol.1, pp.81–106 (1986).

[6] Rosch, E.: Principles of Categorization, in E. Rosch and B. B. Lloyd (eds.) *Cognition and Categorization*, Erlbaum (1978).

# Chapter 4

# Case-Based Decision Support
# Case-Based Diagnosis

# The Application of Case Based Reasoning to the Tasks of Health Care Planning

**Carol Bradburn**
**Department of Computer Science**
**Royal Melbourne Institute of Technology**
**Australia.**

**and**

**John Zeleznikow**
**Department of Computer Science and Computer Engineering**
**La Trobe University**
**Bundoora, Victoria. Australia**
**and**
**Katholike Universitest Brabant**
**Tilburg, Netherlands.**

## extended abstract

This paper describes an application of case based reasoning in the field of health care planning. The process is modelled in the FLORENCE expert system, an experimental prototype which models the reasoning of an expert clinician in advising on the three basic planning tasks of *diagnosis*, *prognosis* and *prescription* within a Nursing domain. We have developed an empirical approach which models the reasoning processes of expert clinicians. Both rule-based and case-based reasoning are used where appropriate. It has been found that case-based reasoning is especially appropriate to situations where decisions must be made about the progress of cases over time.

***Diagnosis*** is defined as the process of evaluating health status by making a set of observations. This meaning should be differentiated from the common medical meaning of diagnosis as the identification of the cause of a fault or disease. The diagnostic module of FLORENCE is essentially rule-based being structured around the "health patterns" of Gordon [1]. However, the diagnostic module generates records of cases containing numerical indicators which form the indices for retrieval of suitable cases in the other two modules. The numerical indicators produced represent the health status of a client. The essential diagnostic process is one of reducing the task of evaluating the large, general health patterns of Gordon's model to one of evaluating easily observable client parameters. Repeated observations may then serve as a basis for measuring change of status and/or evaluating the effects of treatments.

Gordon defines a *pattern* as "a sequence of behaviour across time" and delineates eleven areas in which such behaviours may be observed. Examples of these patterns are: *activity-exercise pattern, nutritional-metabolic pattern* and *elimination pattern.* For each health pattern, *sub-concepts* were chosen in consultation with experienced clinical nurses. Figure 1 shows the activity-exercise pattern which has 9 associated sub-concepts. Each sub-concept is related to *critical indicators*; these are the observable parameters that may be assessed by the clinician in regard to a particular client. For any sub-concept, the critical indicators are not necessarily extensive; it being more important to identify those that most accurately predict health status. Figure 2 shows the sub-concept mobility and its associated indicators. Any indicator may relate to one or more sub-concepts ; for any given sub-concept, the indicators may vary in significance.

| pattern: activity-exercise | sub-concept: mobility |
|---|---|
| sub-concepts:<br>exercise tolerance<br>mobility<br>occupation/recreation<br>self-care<br>home maintenance<br>airways clearance<br>breathing pattern<br>cardiac output<br>tissue perfusion | health pattern : activity-exercise<br><br>indicator weighting<br><br>physical movement 5<br>motor function 4<br>conscious state 3<br>musculo-skeletal dev. 3<br>happiness 1 |
| **figure 1 : the activity-exercise pattern** | **figure 2: the mobility sub-concept** |

The varying significance of the indicators is represented by a weighting system which allocates an indicator a score on a scale of **5** .. **1**, where a score of **5** denotes major significance and a score of **1** denotes minor significance. Essentially, this weighting, answers the question, "to what extent does indicator X predict the health status of sub-concept Y?". Figure 2 also shows the weightings of the mobility sub-concept indicators.

The user enters his/her observations of the indicator in the client. Indicators in a client are assessed by observation as being on a scale of **+2** .. **0**.. **-2**, where **+2** indicates very satisfactory ("well above average"), **0** indicates normal ("average") and **-2** indicates very unsatisfactory ("well below average"). "Average" is taken as meaning the usual level of this indicator in the general population .

When entry of the observation is complete a numerical evaluation of client status is calculated. The client score on an indicator is related to the weighting of that indicator, giving a contribution to the sub-concept status. In turn, each sub-concept score of a pattern may be combined to give a total numerical score for that pattern. In addition to the calculation of these numerical indicators, FLORENCE reports on the presence of any specific problems ("nursing diagnoses") displayed by the client. The diagnostic system, on request from the user, writes to the client record information about:

i. abnormal indicators detected.
ii. nursing diagnoses made
iii. progressive changes within the health parameters.

Repeated observation allows accumulation of progressive data about a client's changing health status. A client record or "case" comprises a series of timed observations together with the derived numerical indicators. A bank of completed client cases provides the basis for reasoning about change over time which is fundamental to the other two tasks - prognosis and prescription.

FLORENCE defines ***prognosis*** as the prediction of changes in health status, simulating the real life activity of "remembering" similar real cases in the past. Briefly, if completed cases can be found that, at some time in their development, were similar to the present new (incomplete) case the new case is extrapolated forward in time on the assumption that it will follow a similar course to the retrieved cases. The process may be considered as having two stages: (i) finding similar cases and (ii) formulating future projection(s).

A major problem in any system using case based reasoning is the *retrieval of cases* from a case base within a reasonable time. It is our aim to avoid the use of abstract indices, rather allowing case features to form their own indices. However, it is obviously undesirable to search every case on a multitude of features. The method used in FLORENCE retrieves, on

an initial pass through the case base, a subset of cases on a single indexing feature; this subset may be expected to contain all relevant cases. This, in itself, is a not inconsiderable task. The FLORENCE prototype has about 30 cases each containing between 7 and 10 days of records. The current prototype is developed in HyperTalk; it is anticipated that a system for use in the real world would need re-engineering to maximise efficiency.

The initial parameter used for case retrieval is the overall health score for a particular day. All cases are retrieved which have any day with a health score "close" to the health score of the current day in the new case. "Close" is experientially defined; we have found that to seek a score within 0.5 of the current day score retrieves a useful, but manageable, selection of cases. The old case name and the day are entered into a list of possible similarities.

The pool of retrieved case-days will contain all similar days. However, it is possible that it may contain case-days that have similar health scores but, as the total health score is an average of concept scores, quite different distributions of concept scores. Therefore, further similarity metrics are applied to the retrieved cases.

For each case-day a *difference score* is calculated. This provides a numerical estimation of the differences between concept distributions of the old case-day and the new case-day. The difference score gives a much closer approximation of similarity between case-days then a simple comparison of health scores. Those case days with the lowest difference scores are most similar.

There is also provision to refine the process further and consider similarity at sub-concept level. Although sub-concept scores are calculated during the diagnostic process, these are regarded as intermediary scores and are not retained in the permanent client record. However, a record is kept of abnormal clinical features along with their commencement and conclusion days; abnormal clinical features relate directly to sub concepts. Therefore, a measure of sub concept similarity may be approximated by calculating the number of abnormal features that old-case-day and new-case-day have in common giving a *feature score* .

On the basis of the difference scores and feature scores, cases can be ranked into order of similarity. A *rank score* is calculated by ranking all difference scores in inverse order, ranking all feature scores in ascending order and taking an average of the two rankings

*Projections* into the future based on similar cases may now be made. Initially it was our intention to make a composite projection based on several similar cases. This may be viewed conceptually as the formulation of a prototypical or "average" case. However, difficulty with the development, and hence usage, of a prototypical case was the confounding effects of different treatments. What we would really like to know from a prototypical case is "what will happen if nothing is done?". However, it is rare to see a case, especially one showing abnormal features, in which some treatment is not given. It is obviously unethical to withhold well accepted interventions in order to see what will happen without them. Therefore, we modified the system to formulate several individual projections. This would also have the effect of allowing comparisons of differing treatments to be made.

The user selects the desired case days, from the pool of ranked days, on which to perform a projection. This user input allows the elective choice, if desired, of early days in an old case; these may not always be the "best" days but are more likely to allow a lengthy projection. In addition, the user may utilise "hunch" type knowledge, which may be based on the memory of specific persons.

The *projection* process commences by comparing the overlapping days (the current day and the retrieved old case day). For each health pattern, a correction factor is calculated to align the score of the old day with that of the current. The current case is then projected forwards by applying the correction factors to each pattern of the subsequent old case days. Experiments with te system show that the process well replicates the prognoses formulated by an experienced clinician. When used to project on real cases in which the outcome is known, close matches do occur but, in a minority of cases, the outcome is quite different from that projected; this is to be expected as a result of the large number of variables that may affect human health status. The function of the prognosis module may be mainly viewed as one of asking "what if?" questions about case development.

***Prescription*** is defined as implementing interventions that affect the environmental factors impinging on the health status in such a way as to influence health change in a positive direction. The effects of treatments can only be reasonably determined retrospectively by noting changes in observations. However, it is obviously important to predict in advance which treatments are likely to be useful for a particular client. Again, this may be performed by utilising expert knowledge about the likely uses and effects of treatments. FLORENCE actually incorporates an essentially rule-based adviser which groups treatments relevant to promotion and cure in different health areas. This is seen as inadequate because of the many treatment choices and combinations of choices within and between health areas. We have, therefore, developed an advisory module which suggests appropriate treatments on the basis of experience in real past cases.

The input to this prescription advisory module consists of the single case projections developed in the prognosis module. The process is one of selecting the best treatments for each health concept by considering individual health concept outcomes in each case. This provides a pool of treatments which have been shown to be effective in similar cases. The problem here is that, because these treatments are derived from several different cases, some may be mutually incompatible. Our approach is to examine the occurrence of treatments within their original cases and dynamically derive constraints on their interactions. For example, such constraints may derived as never having two treatments together in the same case or of one treatment always preceding another. With these constraints, the selected "best" treatments are then projected onto a temporal prescription plan which suggests which treatments should be used, at what times and in what combinations.

From pool of "projection" cases the "best" cases are selected for each of the health patterns. A "best" case is the one that shows the greatest improvement in pattern score from the day of projection start until the end of the case. Then for each concept, the treatments that were used in each of the best cases are collected including the commencement and completion times of each treatment. This collection forms the basis of the suggested treatment plan.

Next the embryonic plan is rationalised to ensure that recommendations for conflicting therapies are not made. Therapies may conflict by inappropriate occurrence (eg. prescription of two therapies whose effects negate each other; prescription of multiple therapies of which the cumulative effect is negative) within the same case, or by temporal conflict (allowing therapies to overlap inappropriately or prescribing therapies in an inappropriate sequence).

Certain modifications are made to the plan without the necessity of consulting the case base. Firstly, *rare events* are eliminated; a treatment that is derived from less than half of the contributing cases is defined as a rare event. It is assumed possible that this treatment is uncommon, possibly specific to some unusual case. These rare events are eliminated from the treatment list. Secondly, *duplications* are resolved. It is possible that the same treatment was prescribed in two or more of the source cases. In this case, the treatment is initially retained , commencement and completion times being taken as the mean of those of the

contributing cases for that treatment. Remaining treatments are then sorted in order of commencement time.

The main danger to be considered in a plan developed from multiple sources is that treatments may be given in harmful combinations. FLORENCE handles this problem by searching the full case base to ascertain which treatments remaining in the plan have either never been given together in the same case or have never overlapped in any case. With this derived knowledge, incompatibilities in the plan are resolved. One of two mutually exclusive treatments is eliminated; comparison of surviving treatments is continued until all survivors can be shown to be compatible with each other. Compatible, but non-overlap treatments have start and stop times as necessary to produce compatibility. This means that the final prescription plan will be safe although it may not necessarily be the optimum plan.

The prescription module of FLORENCE is the in some ways the least satisfactory. Even in a prototypical system with a small case base, the search time involved in deriving the exclusion and overlap constraints is considerable. The difficulty is that, for the derivation of accurate constraints, a really large case base is desirable but increasing case base size increases search time. Current work is being undertaken in the development of a sub-system, separate from the prescription module, to "learn" the constraints; the prescription module may then simple look up a record of learned constraints about any treatment. Of some concern also is the usefulness of detailed plan generated for future use in a changing environment. However, the possibility of generating and comparing multiple plans make prescription, as with prognosis, a useful "what if" exercise.

In summary, the FLORENCE system provides advice to the health care planner on the tasks of diagnosis, prognosis and prescription. A feature of the system is the calculation of statistical parameters representing various aspects of health care status. These are stored, along with the observed features and prescribed treatments in a client case which provides a record of the ongoing health profile. These old client cases may then be used to predict likely outcomes and to suggest suitable treatments in a new case.

**reference**

[1] Gordon.M. (1987) *Nursing diagnosis: process and application (2nd edn.)* New York: McGraw-Hill Book Company.

# CASE-BASED REASONING:
# APPLICATION TO THE AGRICULTURAL DOMAIN, A PROTOTYPE

K.C. Chiriatti* and R.E. Plant**

* Metapontum Agrobios, S.S. Jonica 106 km 448.2,
75010 Metaponto (MT) Italy.
** Dept. of Agronomy & Range Science University of California,
Davis

ABSTRACT

A Case-Based Reasoning (CBR) system prototype for managing the use of fertilizers on farm land has been developed, adapting the CBR techniques used in CHEF, the planning system in the cooking domain (Case-based Planning. Kristian J.Hammond. Academic Press 1989). The prototype creates fertilization plans for citrus farms by accessing a case library of citrus-tree fertilization.
It addresses the main issues of the CBR techniques: representing and indexing past cases, retrieving and modifying old plans and explaining and learning from failures in fertilization schedules.
Developing that prototype is a step towards understanding how CBR system could be used to aid humans in solving problem process in the agricultural domain.

## 1. The Domain

Modern intensive agriculture typically uses large quantities of nutrients to achieve high levels of production. When these nutrients escape the agricultural system (e.g.,nitrate leaching), nutrient use efficiency is lowered and pollution may result. A correct fertilization should return the amount of nutrient removed from the plant-soil system during vegetative and reproductive orchard growth. Knowledge of soil properties, soil nutrient availability, climatic conditions, orchard performance and cropping operation is needed for identifying causes of nutritional imbalance and suggesting correct fertilization. Antagonistic and synergistic relationships between plant nutrients must also be taken into account.
In order to make sound fertilizer recommendations, it is necessary to relate the diagnostic indexes to the amount of nutrients required for optimum yields. In many instances, rules-of-thumb as well as experience of the agronomist or extension specialists can provide a useful mean for planning fertilizations. Suggestions from past cases often help citrus experts to define fertilization schedules, by recalling a previous schedule and adapting the old solution to the new scenario in terms of soil properties, nutrients balance, climate and weather forecast.
The fact that citrus experts use experience with previous fertilization plans to define the new ones, makes case-based reasoning particularly appropriate for decision support system in managing and planning fertilization schedules.

## 2. Case-based Planning: the Chef Model and our Prototype

Case-based planning is planning from experience [1]. The basic idea is that a machine planner should use its own experience in developing new plans.
Past successes are recalled and modified to create new plans, memories of past failures are used to avoid having same problems again and past repairs are remainded to solve them [2]. Successful plans are stored in memory, indexed by the goals they satisfy and the problems that they avoid. Failures are also stored, indexed by the features that predict them. By storing fai-

lures as wellas successes, the planner is able to anticipate and avoid future plan failure [3].
This planning theory has been implemented in Hammond's system Chef, which creates new plans, using the old ones, in the Szechwan cooking domain.
Our prototype's goal is to plan a citrus fertilization schedule given a set of slots which describe cultivar, tree age, soil properties, visual symptoms, cropping operation, orchard perfomance and leaf analysis data, if available. We have built a case library containing citrus fertilization schedules. Each case, represented by the MOP (Memory Organization Packed) structure [4], consists of a set of slot describing the citrus orchard features and state, and the planned fertilization schedule expressed in terms of steps. As a recipe the schedule consists of a set of steps that citrus growers should follow to achieve high quality yield and control the fertilizer inputs (fig.1).

```
(DEFMOP I-M-SCHEDULE1 (M-SCHEDULE)
  (CULTIVAR I-M--NAVEL)
  (AGE I-M-20)
  (SOIL I-M-CLAYEY-SOIL)
  (LEAF I-M-LEAF-SURFACE)
  (LEAF-COLOR I-M-YELLOWISH)
  (LEAF I-M-CENTRAL-VEINATION)
  (PART-COLOR I-M-YELLOW)
  (CHEM I-M-LOW-STANDARD)
  (EL-CHEM I-M-N)
  (FRUIT-QUANT I-M-HIGH)
  (FRUIT-SIZE I-M-SMALL)
  (PROD I-M-100-Q/HA)
  (ADVICE M-ADVICE-STEPS
       (DEF-AM-STEPS M-STEP-GROUP
        (1 M-DEF-AM-STEP (OBJECT I-M-NITRATE)
                         (QUANTITY I-M-200-KG/HA)))
       (SPLIT-AM-STEPS M-STEP-GROUP
        (1 M-SPLIT-AM-STEP (DOSAGE M-AMOUNT-GROUP
                           (1 I-M-100-KG/HA)
                           (2 I-M-50-KG/HA)
                           (3 I-M-50-KG/HA))))
       (SPLIT-TIME-STEPS M-STEP-GROUP
        (1 M-SPLIT-TIME-STEP (DATE M-DATE-GROUP
                             (1 I-M-APRIL)
                             (2 I-M-15JUN-31AUG)
                             (3 I-M-15JUN-31AUG))))
       (TREAT-STEPS I-M-EMPTY-GROUP)
       (IRRIGATION-STEPS I-M-EMPTY-GROUP)))
```

Fig.1 A case from the memory.

Like Chef's architecture [1], our prototype is composed of processes and knowledge structures, it consists of the modules: anticipator, retriever, modifier, repairer, storer and the assigner. Goals are handed to the anticipator, which tries to predict any problems that might arise while planning for them. If a problem is predicted, a goal to avoid it is added to the initial goals. The retriever identifies the most appropriate cases in the case memory and presents them to the modifier. The simplest way to identify the most similar case is to use nearest-neighbor search [5]. But this is an expensive operation and its cost grows with the size of the case base. To avoid such an exhaustive comparison without compromising accuracy, we have organized cases in memory using indices, generally the most discriminating features of the cases. The retriever compares only the indices with the new

problem and retrieves those cases whose indices match the new problem. To improve the performance of this process, indices are organized hierarchically [6]. After retrieving cases based on indices, the retriever searches the best match comparing the new problem features with those of the case instances under the matched index.
The retriever can only find and suggest past schedule for new situations: it cannot do anything about modifying these schedules. The modifier adapts the retrieved schedule to satisfy goals not already satisfied. In order to adapt the old schedule the modifier needs knowledge about the soil properties, how and how much nutrients can be uptaken by the different soil type. For example, if the soil type in the retrieved case was a sandy soil, but the current soil type is a sandy-loam the nitrate amount is splitted in different steps and the dose is calculated by a formula. The modifier to handle the changes uses a library of modification rules and domain knowledge that outlines how it should adapt specific domain features while using its more general rules.
The built plan is run and the results are checked against the planner's initial goals. The planner runs a schedule simulation and uses the results to diagnose errors. It can even ask an outside source if the plan behavior is what expected. In that way some unpredictable events as rainfall or insects pressure can be taken into account when a fertilization is planned. The plan simulation is done by using a model. The model [7], considering climate data (rain, temperature, evaporation), water holding capacity of the current soil type and information related on plant phisiology, estimates the chemical losses and verifies if the nutrient amounts in the modified schedule return the plant requirements. Checking whether the defined schedule is a success or a failure means to verify that the initial goals have been satisfied.
Thus, the system searches if there is a goal violation in the events chain occured during the simulation and then goes backword to identify the step that caused the failure, if any. Successful schedule is handed to the storer and placed in memory indexed by the same features that will be used to access it. The indices used to store schedules are satisfied goals. If there is a failure the plan is given to the repairer that builds a causal description of why a fertilization has failed. The explanation pointing to the actions that caused the failure, provides the focus as to what parts of a plan have to be changed. The system uses the explanation to find a structure in memory that organizes a set of strategies for solving the problem.
These structures called TOP (Tematic Organizatin Packed) [4] are general indices for a set of repair strategies. For example, even though Potassium and Magnesium were given in the right amount, a deficiency of one could result. The deficient nutrient should not be given to the plant, being present in the soil, because the interactions between those nutrients prevent the tree to uptake both. The explanation leads to the TOP labeled: side effect disabled condition concurrent [3]. Under that TOP there is the repair strategy that will fix the deficiency by deleting the schedule step in which the deficient nutrient is suggested to be given. The repaired schedule will be placed in memory indexed by the avoided problems as well as the achieved goals. While the repairer is fixing the schedule, the assigner decides wich features caused the failure. Then it can extrapolate from these to the features in later situations that could arise again. The assigner's output is not a plan, but is a knowledge base of possible problems that can arise and the circumstances that predict them. Its output can be a set of inference rules that are fired in the early stages of planning (e.g., if it is winter time, no nitrate should be used because the rain may leaching it). The anticipator module which anticipate problems on the basis of features marked by the assigner has to take these rules and making predictions before other planning is done.
We are still implementing the assigner and anticipator modules.

## 3. Results and Discussion

When we started this work, we examined different case-based systems [8] and determined that, of available case-based reasoning paradigms, that developed by K. Hammond for the Chef system would be most effective for agricultural domain.

Working on the citrus fertilization planner we determined the key ways in which the agricultural domain differs from the recipe domain of the original Chef program, and the modifications necessary to adapt Chef to our domain.

Principal among these differences are: 1) the fact that in the agricultural domain the outcome of a plan may not match that expected due to unanticipated effect (e.g.,heavy rainfall, fungie or insect pressure, etc.), and modifications to planner must be made to help to anticipate these effects or at least respond them in future situations; 2) modify the adaptation process so that pieces of different old schedules that partially match the new ones can be used.

The prototype development demonstrated how case-based reasoning and a fairly large case base can be used for planning agricultural crop management. Much work still remains.

Our first priority is to build a full system. We also need to investigate the effectiveness of simulation models and the similarity assessment among the retrieved cases. Finally, we plan to integrate the case-based system with a rule-based expert system to create a useful decision support system for extension specialists and citrus growers.

## References


1. Hammond, Kristian J. Case-based planning: Viewing planning as a memory task. Academic Press,Inc. 1989.

2. Hammond, Kristian J. Case-based planning: a framework for planning from experience. Cognitive Science,1990,14, 385-443.

3. Hammond, Kristian J. Explainingand repairing plans that fail. Cognitive Science,1990,45,173-228.

4. Schank,R.C. Dynamic Memory: A theory of learning in computers and people. Cambridge University Press, Cambridge 1982.

5. Kolodner J.L., Simpson Jr. and Sycara-Cyranski. Aprocess model of Case-based reasoning in problem solving. Proc. Int'l Joint Conference Artificial Intelligence, Morgan Kaufmann, San Mateo, CA 1985, 284-290.

6. Duda R.O. and Hart P.E. Pattern Classification and Scene analysis. John Wiley and Sons, N.Y. 1973.

7. Follett R.F.,Keeney D.R. and Cruse R.M. Managing Nitrogen for groundwater quality and farm profitability. Soil Science Society of America, Inc. Madison, Wisconsin, Usa, 1991.

8. Riesbeck,C.and Schank R. Inside case-based reasoning. Lawencer Erlbaum Associates,Inc. Hillsdale N.J. 1989.

# Using CBR techniques to detect plagiarism in computing assignments

Pádraig Cunningham

Department of Computer Science, Trinity College Dublin

Alexander N. Mikoyan

Dept. of Mathematics and Theoretical Mechanics, Moscow State University

**Abstract.** The problems of case retrieval in CBR and plagiarism detection have in common a need to detect close but not exact matches between exemplars. In this paper we describe a plagiarism detection system that has been inspired by ideas from CBR research. In particular this system can detect similarities between programs without performing exhaustive comparisons on all exemplars. Our analysis of similarity in this well controlled domain offers some insights into the kinds of profiles that can be used in similarity assessment in general. We argue that the choice of a perspicuous profile is crucial to any classification task and determining the best predictive features may require significant analysis of the problem domain.

## 1 Introduction

The problem of detecting plagiarism in computing assignments depends on being able to identify similar programs in large populations. This emphasis on similarity, on identifying close matches, is reminiscent of the problem of case retrieval in CBR. In this paper we will concentrate on the application of CBR techniques in Cogger*, a system for detecting plagiarism. We will discuss what this novel domain informs us about retrieval in CBR and about the automatic assessment of similarity in general. Our considerations on similarity in this well controlled domain offer some insights into the alternatives of statistical and knowledge based classification.

Since the idea of similarity can be considered along several dimensions it is often difficult for humans to agree on when cases, or programming assignments, are similar. In this research the programs under consideration have complicated structure and programs are considered to be similar if their function call structure is similar. This involves the determination of the similarity of function call trees; the mechanisms we use are described in Appendix I.

Before examining the problem of plagiarism for a CBR perspective we will introduce some theoretical issues in CBR that are relevant. In section 3 we discuss similarity in general and in section 4 we consider the issue of problem representation that must be considered before any similarity can be determined. We believe that a basic tenet of the majority of CBR research is that similar cases can be retrieved from the case-base inexpensively; in section 5 we consider what kinds of representations are required to support this.

## 2 Theoretical Issues

Currently in AI there is a view that knowledge representation is unsuccessful and knowledge acquisition is fraught with problems. Consequently there is a move towards an AI paradigm that avoids these issues. This new AI is based on statistics and weights rather than symbolic knowledge representation [1]. The current popularity of connectionism is evidence of this. Closer to CBR, Memory Based Reasoning (MBR) is an approach to AI that wishes to avoid knowledge acquisition and domain modelling [2]. The great attraction of neural networks and MBR is the contention that expert performance can be achieved without knowledge level analysis of the problem domain. This is in sharp contrast with the conventional view in AI; the view that "In the knowledge lies the power" and the knowledge must be represented explicitly.

CBR is a methodology that can serve both of these paradigms. Case-Based Reasoning systems can be **information theoretic** or **knowledge-based**. CBR systems for simple tasks like diagnosis or property valuation can be set up with little analysis of the problem domain. At the other end of the spectrum systems for more complex tasks like design require a complex domain model in order to process retrieved cases.

The main theme of this paper is the implications that these issues have on determining similarity in case retrieval. Is it possible to establish the similarity of two cases in a system that does not have a strong domain model? How far can we go with shallow index features in case retrieval? To this end we will analyse similarity in the context of detecting plagiarism in computing assignments. This is not really a CBR problem but we will argue that the issues of similarity are the same nonetheless.

* "Cogging" is an anglo-irish slang word for copying homework or other exercises.

## 3 Similarity in CBR and in Plagiarism Detection

The standard approach to the problem of detecting plagiarism is to produce a profile reflecting the use of keywords and identifiers and to use this signature to produce a table describing the 'distance' between different programs (see [3][4][5] for instance). Research in CBR and machine learning has developed methods for assessing similarity in large populations that are more sophisticated than this so in this paper we apply CBR insights on similarity to the plagiarism detection problem.

In CBR the objective in noticing similarity is to allow for reuse of old solutions in new situations. In plagiarism the object is to identify similarity that betrays a common origin. In CBR similarity may be based on surface features or on features that are more abstract and structural. In plagiarism detection attempts will have been made to conceal superficial similarity and detection must be able to identify systematic or structural similarities.

### 3.1 A Brief overview of CBR

In attempting to apply CBR techniques in this domain our understanding of the stages in CBR are as follows:-

- Case Representation
- Case Indexing/Retrieval
- Mapping
- Adaptation

A fundamental idea in much of CBR research is that the identification of the best matching case from the case-base should be a two stage process [6]. **Base filtering** is the first stage where a set of candidate cases are selected from an indexed case-base. The case-base will often be organised as a discrimination net to facilitate this. The second stage (**Case Selection**) will select a case from this candidate set based on a more detailed comparison of the cases. A mapping between the base and target cases may also be produced at this stage. Such a selection that does not involve an exhaustive search of the case-base is a novel idea in plagiarism detection.

For plagiarism detection we will concentrate first on producing a representation of the programs for mapping, before analysing the mapping process itself. More than anything this exercise in plagiarism detection highlighted the importance of this parameterisation process.

## 4 Representation

The first phase in the development of a CBR system involves deciding on a representation of the cases in the system (Fig. 1). This phase is crucial because the perspicuousness of the representation greatly influences the success of the subsequent phases. It is important to note that the contribution of neural networks in tasks of this type is in the classification process that operates on the chosen representation (see the mushroom classification task in [7] for instance. However we are arguing that the crucial phase in this problem solving process is choosing the correct representation in the first place. Indeed given a good predictive representation it may be possible to classify inputs using traditional cluster analysis techniques.

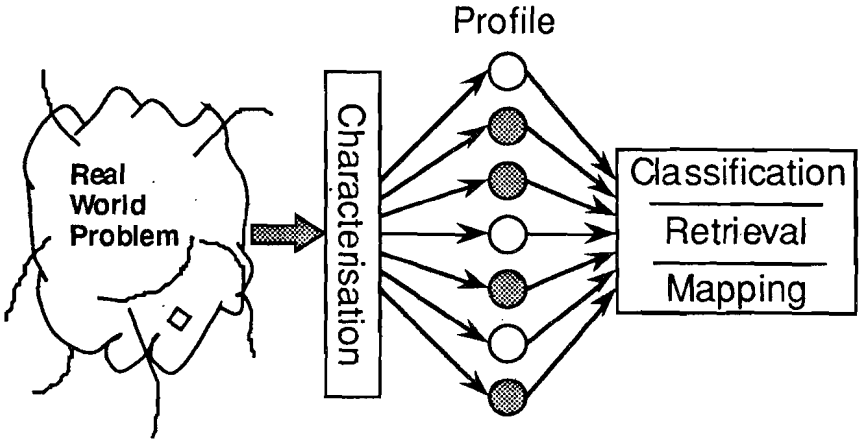



Fig.1. Characterisation involves producing a profile that represents real world problem for use in classification, etc.

From a problem solving perspective it is useful to characterise indexing features along a continuum from shallow to deep (Fig. 2). Shallow features are the obvious surface features of a case and can be determined without much analysis. Deep features are more predictive in the context of the problem in hand but require more analysis to determine. It should be clear that when two cases are very similar they will share surface features. However, when the similarity is more abstract shallow indexes may be different and the similarity may only be indicated in more abstract features. We believe that the plagiarism detection problem that we will describe offers some useful insights in this regard.

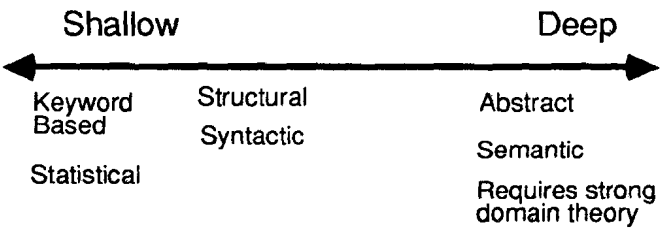



Fig.2. Indexing features can be shallow or deep depending on their semantic content.

## 5 Base Filtering

What we require form base filtering is a computationally inexpensive means of selecting a small set of candidates from the case-base that are likely to be similar to our target case. This is essentially a classification problem and as such there are several different approaches that can be adopted. One important criteria in categorising techniques is whether classification involves an exhaustive search of the case-base. Nearest neighbour techniques are of this type and involve comparing the target case with each base case in turn using a global distance metric. In these cases retrieval is O(n) where n is the number of cases in the case-base. More promising case retrieval techniques structure the case-base as a decision tree and retrieval is O(log (n)) since it does not involve visiting every case in the case-base.*

The most common means of supporting base filtering in CBR is to organise the case-base as a decision tree that will support rough remindings without the need for exhaustive comparisons. Two approaches were considered for Cogger:-

- **Information theoretic**: Using Gennari's Classit algorithm [8]. This method takes a case represented as a vector of numeric attributes and can incrementally locate the case in a classification hierarchy. In case retrieval this classification can produce the candidate set required from base filtering. The main advantage of Classit is that it is incremental, unlike other cluster analysis techniques in statistics.

- **Knowledge based**: Discrimination networks (D-nets). The indexing features are ordered according to importance and the case-base is structured as a taxonomy based on this feature ordering. Cases are classified by locating them in this taxonomy.

D-nets have the advantage that, with redundancy, they can support different types of remindings. However, the use of D-nets commits the user to a knowledge level evaluation of the problem domain and to an ordering of the indices to reflect their priorities. For this reason Gennari's incremental clustering algorithm was used to perform the base filtering in Cogger.

Experiments with this base filtering indicate that no cases are slipping though the net. In no situation has the base filtering failed to capture known similar cases in the candidate set.

### 5.1 Experiments in Cogger

One of the very simplest profiles that can be used as the basis for similarity assessment in plagiarism detection is an ordered frequency count of identifiers in the programs. Matching is done based on comparing ranked frequency counts. Table 1 shows an example of two such profiles. This profile is at the very shallow end of the continuum shown in Figure 2. It is easy to construct without any knowledge of what is going on in the program. This profile is adequate for spotting blatant plagiarisms as is the case here. However, a few simple changes to the programs will fool any similarity metric based on this profile.

* There are also connectionist techniques for classification and similarity that have not been considered here. In particular spreading activation and constraint satisfaction have been used in ARCS [9] and SAARCS [10]. These are localist connectionist systems and are different to other approaches mentioned in this paper.

Table 1. Frequency counts of identifiers in two programs

| Program stu1 | | Program stu3 | |
|---|---|---|---|
| 19 | close | 20 | close |
| 11 | pipea | 10 | pipea |
| 10 | pipeb | 10 | pipeb |
| 4 | hold | 4 | num |
| 4 | dup | 4 | dup |
| 3 | wait | 3 | wait |
| 3 | fork | 3 | fork |
| 3 | execlp | 3 | execlp |
| 2 | squasherpd | 2 | squasherpd |
| 2 | readerpd | 2 | readerpd |
| 2 | pipe | 2 | formatterpd |
| 2 | formatterpd | | |

A rudimentary understanding of the programming language suggests a improved profile based on counts of reserved keywords only. This profile better reflects the actual structure of the program since keywords like `do` and `if` indicate specific control structures. Similarity metrics based on this profile can spot less obvious similarities.

Table 2. Profiles based on counts of reserved identifiers.

| Program sta05 | | Program sta09 | |
|---|---|---|---|
| 4 | char | 2 | char |
| | | 1 | do |
| 38 | else | 41 | else |
| 1 | float | 1 | float |
| 19 | for | 14 | for |
| 52 | if | 56 | if |
| 4 | int | 9 | int |
| 21 | return | 22 | return |
| 1 | sizeof | 1 | sizeof |
| 2 | struct | 2 | struct |
| 22 | void | 27 | void |
| 9 | while | 8 | while |

In the terms introduced in Fig. 2 this is an improved surface profile or a rudimentary structural profile. The profile that we actually used in Cogger is a further improvement on this. Some of the less predictive keywords have been dropped and some structural parameters of the programs have been included. For instance; the first parameter (`top width`) is the number of function calls from the top level of the program, the `depth` is the maximum depth of function calls.

Table 3. Profiles based on structural paramters and counts of reserved identifiers.

| Program sta05 | | Program sta09 | |
|---|---|---|---|
| 39 | top width | 9 | top width |
| 6 | depth | 6 | depth |
| 39 | max width | 31 | max width |
| 13 | user defined | 14 | user defined |
| 13 | system def. | 12 | system def. |
| 0 | recursive | 1 | recursive |
| 0 | do | 1 | do |
| 19 | for | 14 | for |
| 52 | if | 56 | if |
| 9 | while | 8 | while |
| 21 | return | 22 | return |
| 38 | else | 41 | else |
| 0 | case | 0 | case |
| 0 | switch | 0 | switch |

## 6 Evaluation and Conclusions

We have tested these profiles on three data sets containing from 6 to 35 program profiles. When programs are very similar it shows up in all profiles as would be expected. When the plagiarism is more subtle and surface features have been altered the similarity is not evident in the surface profile but shows up in the structural profile

(see Table 3). Determining the appropriate features for this profile required some analysis. So we conclude that for more difficult classification tasks surface profiles are not adequate and the more abstract profiles are more expensive to set up.

From the perspective of the plagiarism detection task, the main novelty of this system is that it operates without doing exhaustive comparisons. Cogger performs similarity assessment as a two stage process; the first stage uses the Classit algorithm [8] to produce the candidate set of cases. The second stage performs expensive comparisons of the program structures to produce a metric of similarity (see Appendix I for details). The main conclusions from this exercise are as follows:-

- Effective profiling is crucial: no amount of cleverness in matching and retrieval can compensate for poor case representation. Figure 1 depicts the characterisation process that produces the regular case representation for classification etc. Settling on the best predictive features was a non-trivial task in Cogger and would be expected to be more difficult in less formal domains.

- Cogger performs very well at identifying similarities in programs; however, it must be acknowledged that using the function tree structure as the basis of the mapping process is crucial to this success.

## References


1. Stanfill C., Waltz D., "Statistical Methods, Artificial Intelligence, and Information Retrieval", in *Text-Based Intelligent Systems*, P. Jacobs ed., pp215-225, Laurence Earlbaum Associates, Hillsdale, New Jersey, 992.

2. Stanfill C., Waltz D., "Toward Memory-Based Reasoning", *Communications of the ACM*, pp1213-1228, Vol. 29, No. 12, December 1986.

3. Grier S., "A tool that detects plagiarism in PASCAL programs", *SIGCSE Bulletin*, pp15-20, Vol. 13, No. 3, February, 1981.

4. Wise M., "Detection of similarities in student programs: YAP'ing might be preferable to Plague'ing", *SIGCSE Bulletin*, pp268-271, Vol. 24, No. 3, March, 1992.

5. Whale G., "Identification of program similarity in large populations", *The Computer Journal*, pp140-146, Vol. 33, No. 2, 1990.

6. Owen S., *Analogy for automated reasoning*, Academic Press 1990.

7. Carpenter G.A., Grossberg S., Reynolds J. H., "ARTMAP: Supervised Real-Time Learning and Classification of Nonstationery Data by a Self-Organizing Neural Network", *Neural networks*, Vol. 4, pp565-588, 1991.

8. Gennari J. H., Langley P., Fisher D., "Models of incremental concept formation", *Artificial Intelligence*, pp11-61, Vol. 40, September, 1989.

9. Thagard P., Holyoak K.J., Nelson G., Gochfeld D., "Analog Retrieval by Constraint Satisfaction", *Artificial Intelligence*, pp259-310, Vol. 46, 1990.

10. Lange T.R., Wharton C.W., Melz E.R., Holyoak K.J., "Analogical Retrieval within a Hybrid Spreading-Activation Network" in *Proceedings of Connectionist Models Summer School*, Carnegie Mellon University, D. Toretzky, G. Hinton, T. Sejnswski Eds, Morgan Kaufmann, 1988.

## Appendix I

Ultimately, the similarity of two programs is judged on the amount of common structure in their function trees. Computationally, this is a difficult task, increasing rapidly with the size of the trees. The solution adopted in Cogger is to convert the tree structure to a string representation and find common sub-strings. This string matching is complicated by what the strings represent and the character of the similarity that should be detected. Consider the following example C program:-

```
#include <stdio.h>
#include <math.h>
float pi = 3.1415926;

void do_it(void)
{
 printf("Hello, world\n");
 printf("cos(pi/4) = %f\n",cos(pi/4));
}
```

```
main()
{
 do_it();
 if (1)
  printf("END \n");
 else
  fprintf(stdout,"END\n");
}
```

This programme has the tree structure shown in Figure 3.

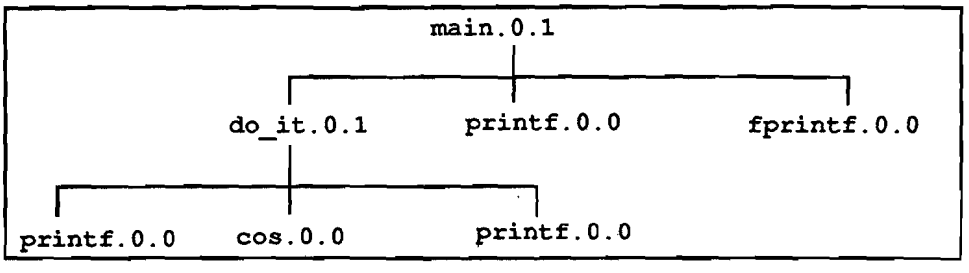



Fig.3 A typical function call structure shown as a tree.

This is converted to the following string format for processing:-

main.0.3.0.0 -- do_it.1.3.0.1 -- printf.2.0.0.0 -- cos.2.0.0.0 -- printf.2.0.0.0 --printf.1.0.0.0 -- fprintf.1.0.0.0

In this format each node has three attributes; its level in the tree, the number of branches, depth of recursion, a flag to indicate whether it is user defined or not. This string representation is used in the matching process; the measure used is as follows:-

$$\frac{\sum \text{matched substrings}}{\text{total length}}$$

Nodes are considered to match if they are user defined and match on the recursion and branching flags or if they are system defined and have an identical name. Substrings match only if the change in the level attribute between nodes is consistent.

# Case-Based Learning of Dysmorphic Syndromes

Carl Evans
Department of Computer Science,
University College London,
Gower Street,
London WC1E 6BT,
United Kingdom.
email: cevans@uk.ac.ucl.cs

## Abstract

The aim of this research is to develop a case-based system to provide decision support for diagnosis of cases of dysmorphic syndromes, and increase the scope of syndrome analysis with respect to rare cases through its learning capability. An interactive case-based model has been designed to facilitate diagnosis through classification, and learning through reorganisation.

Dysmorphic syndromes describe morphological disorders and patterns of morphologic defects[1]. An example is Down Syndrome which can be described in terms of characteristic clinical and radiographic manifestations such as mental retardation, sloping forehead, a flat nose, short broad hands and generally dwarfed physique [4]. There are currently about two thousand registered syndromes which affect about 7 in 1000 children. The primary goal of medical specialists is to diagnose a patient to a recognised syndrome. Firm diagnosis enables prediction of abnormal developmental processes. This may promote a course of treatments that reduce the effects of the disease, or allow genetic counceling to be given in the case of a hereditary disease. However, firm (or even tentative) diagnosis is not always possible. A case may resemble a syndrome but exhibit (or lack) significant features resulting in sufficient doubt for a firm diagnosis. About forty percent of cases remain undiagnosed with respect to known syndromes. A secondary task concerns analysis of rare cases with a view to establishing new syndrome descriptions along with retrospective analysis (and possible reorganisation) of current syndrome categories. Case-based reasoning (CBR) provides an intuitive model for addressing these performance tasks. Diagnosis is in essence a classification task. Conceptually this may be considered as the problem of locating a specific case within a structured case memory in which generalised cases represent syndrome descriptions. The secondary learning task may be considered a reorganisation problem. This may be facilitated by the dynamic properties of a structured case based memory.

An interactive case-based model has been designed to incorporate these two performance tasks. Such an approach was favoured due partly to the weakness of the domain theory [3] and to offer a realistic scope for system performance. Both diagnosis of a case and recognition of a new syndrome or pattern of malformations are highly subjective and ultimately require clinical or radiological investigation rather than symbolic comparison. The scope of the case-based system is to assist in researching similarly affected cases and to focus the attention of an expert towards possible reorganisations. Two particular areas of interest have presented themselves with respect to CBR theory and have greatly influenced the design of this model. The first concerns the actual structure of syndrome categories as they are currently understood which has ramifications upon representational issues of memory and cases along with the subsequent indexing mechanism. The second concerns the general concept of similarity which in this domain is complex and can involve a number of competing factors.

The domain theory is weak in terms of well defined hierarchical categories. Some recognised syndrome families exist, but many syndromes (and cases) stand isolated. Categories are not necessarily clearly

[1] A syndrome can be defined as a pattern of multiple anomalies thought to be pathogenetically related.

disjoint, or mutually exclusive. Standard CBR indexing mechanisms that rely on hierarchical memory structures cannot therefore be employed. Another consideration, which also affects similarity, is that experts interpret case features differently and case descriptions vary in detail. Non-uniform featural descriptions affect both representation and similarity with respect to learning [2]. A distributed approach has been adopted for case memory representation. A memory item (case or syndrome) is a structured object with nodes representing only those abnormal clinical or skeletal regions that are described. In this way the overall shape to the item will vary according to the abnormalities that exist with respect to clinical and skeletal regions. Each type of clinical and skeletal object is stored relative to other objects of the same type (from different memory items) within memory. This fragmented representation has a number of advantages. Firstly it caters for non-uniform data as objects are only instantiated for those features described in the medical record. Whether a case is described by 30 features or 3 features will be reflected in the shape of the resulting structured object. This representation facilitates confined search without the requirement of a hierarchical memory structure. Once indexes have been chosen they can probe the respective regional clusters without the need to traverse a network structure. A further aspect under consideration when choosing the representation was similarity assessment. As described below matching does not necessarily involve an overall assessment of similarity and only a small number of regions may be involved. The structured nature of the representation aids such focus.

Of major interest is the similarity assessment performed by medical specialists. Similarity of cases can be biased by a number of different influences. In one sense similarity assessment may be considered as goal or task driven. Experts often have an initial diagnosis in mind which causes them to focus on a small set of similar features around certain regions of the body rather than examining the overall match. During this time dissimilarities may be regarded as insignificant to the degree that they will not count against a match (ie, overlooked provided they carry minor significance). In contrast if the goal is retrospective analysis of a category with respect to a new case, focus will be on dissimilarities and the interpretation of their respective significance may increase. Feature interdependencies can also be an important factor. These may comprise of combinations of abnormalities that simply appear to commonly occur, or are known to be radiologically or clinically interdependent. Matching may have a number of different temporal aspects and the temporal status (ie, if the patient is alive, or if dead the age of the patient at death) can be very diagnostic. Temporal development is important in some syndromes, for example in Noonan Syndrome clinical appearance and characteristic features change significantly with age. The temporal development of bone and cartilage is important for matching cases of skeletal dysplasias[2] which form a relatively well understood subset of syndromes.

An initial similarity metric utilised domain knowledge of feature weights (from the London Dysmorphology Database [6]). These weights reflect the diagnostic significance of abnormal features. However, significance can vary depending on the goal of the matching process ie, indexing or general similarity assessment. For example, mental retardation occurs in over 600 syndromes and so does not provide a good retrieval index. However, in terms of diagnosis against some syndromes it is a vital feature and so for general similarity assessment may have high significance. This aspect of matching would lead to problems in an unsupervised system in which indexing and matching operate in tandem. The interactive model separates indexing from general matching and allows the user to adjust feature weights to his interpretation of significance with respect to the current group under analysis and the goal of the matching process.

There would seem to be a trade off between the utilisation of a numerical similarity metric based on feature sets and weights ([1]) against a difficult and time consuming elicitation of detailed matching knowledge and cognitive processes. This research has so far favoured a generalised set theoretical approach to similarity assessment [5]. A model of similarity has been designed based on the general matching principles described above that relate to this domain. It establishes a number of different operators that may play a part in overall matching to varying degrees. A matching operator exists for the performance goal ie, whether the performance task is diagnosis or retrospective analysis. This will influence which item (of the two being matched) forms the *subject* and which forms the *referent*. This idea coincides with Tversky's opinion that similarity can be directional, or asymmetric. The weights of similarities and dissimilarities are increased or decreased according to the direction of similarity as-

[2] A dysplasia is an abnormal organisation of cells into tissue(s) and its morphological result(s).

sessment which is in turn defined by the performance goal. Operators also exist to focus on temporal matching, interdependencies and for matching normal regions (currently syndrome descriptions only incorporate abnormalities, but normal regions can be significant in matching).

In conclusion, practical issues have guided the system design towards an interactive case-based model. Interactive control is provided to the user to allow flexibility in retrieval and reorganisation of the case memory, and to offer an aid rather than a solution. The user can control indexing through adjustment of feature weights to account for his own interpretation of significance, and allow him to account for featural equivalences due to the non-uniformity of case descriptions. Following the application of a generalised similarity model to the retrieved memory objects the user is prompted to analyse the similarity mappings produces by the system in order to either confirm a diagnosis or accept (or reject) a proposed link between memory objects or a reorganisation.

## Acknowledgments

This research has been possible due to the assistance of Dr R Winter and Dr J A Maat-Kient (Institute of Child Health, London), Prof D L Rimoin and Dr R Lachman (Ceders-Sinai Medical Center, Department of Pediatrics, UCLA), and Dr C Hall (Great Ormond Street Hospital for Sick Children, London).

## References

[1] David W. Aha. Case-based learning algorithms. In *Proceedings of the DARPA Case-Based Reasoning Workshop*. Morgan Kaufmann, 1991.

[2] Ray Bareiss and James A. King. Similarity assessment in case-based reasoning. In *Proceedings of the DARPA Case-Based Reasoning Workshop*. Morgan Kaufmann, 1989.

[3] Bruce W. Porter, Ray Bareiss, and Robert C. Holte. Concept learning and heuristic classification in weak theory domains. *Artificial Intelligence*, 45, 1990.

[4] Hooshang Taybi and Ralph S. Lachman. *Radiology of Syndromes, Metabolic Disorders, and Skeletal Dysplasias, 3rd Edition.* Year Book Medical Publishers, Inc, 1990.

[5] Amos Tversky. Features of similarity. *Psychological Review*, 84(4):327–352, 1977.

[6] R. M. Winter, M. Baraitser, and J. M. Douglas. A computerised data base for the diagnosis of rare dysmorphic syndromes. *Journal of Medical Genetics*, 21:121–123, 1988.

# Facilitating Sales Consultation through Case-Based Reasoning

**Achim G. Hoffmann*, Sunil Thakar**
Daimler-Benz AG
Research and Technology
Systems Technology
Alt-Moabit 91b
Berlin 10559, Germany
email: achim@DBresearch-berlin.de
email: thakar@DBresearch-berlin.de

**Abstract.** In this paper, we not only describe the sales advisory system - which uses case based reasoning - but also how we tackled both issues of indexing and case adaptation in a novel, intelligent and effective way. Here we propose a multidimensional indexing technique, which is capable of incremental adaptation to the requirements derived from newly occurred cases. We also suggest and advocate in favor of composition of multiple cases or parts of cases. A solution is composed from multiple cases which are similar to the new case with respect to different indexing dimensions. The developed case-based reasoning technique for adapting case indexing is multidimensional and generic in nature. Furthermore, we also provide an analysis of the cognitive task of sales consultation.

## 1 Introduction

In present business environment especially in the manufacturing industry the functions of sales organizations have become complex as products are becoming multi-variant and customer requirements(wishes) high and specific [11]. Due to the increased complexity of sales situations a strong, competent and efficient sales consultation is often required. Thus to facilitate and accelerate the sales consultation process the advisory systems are of utmost importance. Sales consultants usually improve their skills through practice. During the sales dialogue, consultants often use the experience from previous consultation sessions, which help improve their understanding of the needs of the new customer. Moreover, usually first the consultant tries to clarify the most important points for a decision and later to suggest trade-offs among less important points, if necessary. Hence, a case-based reasoning approach to sales consultation appears promising, although the known techniques seem to be insufficient for adequately treating the consultation task.

In this paper, we not only describe the sales advisory system (in section 3) - which uses case based reasoning - but also how we tackled both issues of indexing and case adaptation in a novel, intelligent and effective way. Here we propose a multidimensional indexing technique, which is capable of incremental adaptation to the requirements derived from newly occurred cases. We also suggest and advocate in favor of composition of multiple cases or parts of cases. A solution is composed from multiple cases which are similar to the new case with respect to different indexing dimensions (see section 2). The developed case-based reasoning technique for adapting case indexing is multidimensional and generic in nature. Furthermore, we also provide an analysis of the cognitive task of sales consultation.

### 1.1 The Problem Domain: Sales Consultation

In this context, let us consider the following scenario:

#### Scenario

A customer presently owns a small transport company and is close to exceeding the load/volume capacity of his present fleet. He can increase his load/volume capacity by either:

*Author's present Address: School of CS & Eng. University of New South Wales, P.O. Box 1, Kensington, NSW 2033, AUSTRALIA, e-mail: achim@cs.unsw.oz.au

1) Buying/adding new trucks or 2) Leasing some more trucks or 3) A sensible combination of two or 4) Reorganizing his present activities + buying/leasing new trucks.
Consider a situation, where he still is not able to find an optimal mix on his own. In this situation what he requires is an able consultant.
He walks into a local sales office of a leading truck manufacturer. After the pleasantries, sales representative proceeds systematically; querying the customer to determine the customer's requirements, then mapping requirements to the actual product/component leading finally to a product. While doing so the sales person generally falls back upon the prior sales cases.

### 1.2 Desiderata for a Sales Consultation System

A close look at the scenario above shows the complexity involved in such decision making situations. The question is not simply to optimize some specific objective but any proposed solution will be the result of balancing competing goals. There are not only technical issues, but financial and organizational objectives should be considered as well. Merely a solution would be insufficient; rather explanations are required to persuade and convince the decision maker that the proposed solution is reasonable.
The present situation in this field is that the sales person does everything from requirements analysis to product configuration; from present organizational & financial situation analysis to suggesting an appropriate solution etc. manually. This task requires an enormous amount of knowledge and experience of a sales person in various subfields ranging from product component/ configuration knowledge to financial marketing etc. Todays ever changing product development and financial market situations do not allow all sales person to have the same degree of experience and knowledge about every subfield involved in the decision making. Often, a proposed configuration is technically impossible, which is figured out by the technical staff and results in unpleasant additional costly consultation sessions. A computer support in this situation can be of great help.
The motivation for the current work is not only to facilitate and accelerate the sales consultation process but to improve upon the quality of the consultation as well. We mean quality of the sales consultation in terms of a large number of alternative solutions considered in less time and in terms of the outcome of the consultation i.e. how well does the sales object offered to the customer fits into his/her environment or to his/her specific needs.

## 2 A CBR approach to sales consultation

Case-based Reasoning (CBR) [7] is a method of using previous episodes to suggest solutions to new problems. CBR allows a reasoner to solve problems efficiently when previous similar experiences are available. Problem solving using case-based reasoning usually involves retrieving relevant previous cases, adapting the solution(s) from the previous case(s), if necessary, to solve the problem, and storing the current episode as a new case to be used in the future [9].
Kolodner (in [8]) distinguishes two styles - problem-solving and interpretive - of CBR. In the problem-solving style of the case-based reasoning, solutions to new problems are derived using old solutions as a guide. CBR of this type supports a variety of problem-solving tasks, including planning, diagnosis and design. In the interpretive style new situations are evaluated in the context of old situations. This style is generally useful for situation classification; the evaluation of solution; argumentation; the justification of a solution, interpretation or plan; and the projection of effects of a decision or plan.
All of the CBR systems developed to date [10] fall under such 'natural' domains like medicine or law, which are historically suited to this style of reasoning. What makes our domain different - and somewhat unconventional - from the traditional domains of CBR is that, in our application no 'exact' solutions exist ('identical' customers may want different solutions) and any proposed solution can always be modified. What is important here is that a good approximation/consensus with the customer, should be reached in a fewer negotiation steps, i.e. the system should produce a short list of high quality suggested configurations.

### 2.1 The domain of sales consultation

The customer's choices of particular options will not be arbitrary or unpredictable, since the choices usually depend on the customer's needs, desires and preferences in terms which are more abstract than technical features (e.g. particular options).
Our approach tries to identify characteristic properties of customer needs which are suitable to determine most of the possible choices in most cases.
One source of knowledge which guides the identification process of the characteristic properties of customer needs are previous cases of sales consultation. However, since the number of cases will be relatively

small compared to the number of possible sales objects (in the case of truck configurations more than $10^{100}$), additional knowledge is necessary which guides the interpretation of the stored cases. This kind of knowledge should determine the assumed interdependencies between the choice of options. In our approach, it defines groups of attributes for which typically fixed interdependencies among their values exist. E.g. for a given color of a car, usually only very few colors for the interior will be desired by a customer, which could be derived from previous consultation cases.
If these assumed interdependencies do not hold in the particular sales case, it does not result in an insufficient choice of the sales object but in a prolongation of the consultation session.

## 2.2 Multidimensional indexing

The rather abstract characteristic properties of the customer needs are used in our approach as case indexing for retrieving similar cases. They are called *high level features (in short HLFs)* and are considered as additional features of the sales objects. I.e. they are supposed to describe how well a given sales object meets certain customer requirements.
In order to retrieve previous sales cases which are similar to the customer's requirements catalogue, the HLFs are used as indices for case-based reasoning. Since the values of different HLFs are partly logically independent, it is useful to index the cases according to multiple criteria.

## 2.3 Composing solutions from multiple cases

As a consequence, the proposed solution is composed from different cases, where each case is similar to the case with respect to another HLF. The HLF which serve as indices for the cases are conceived to be initially provided by an expert, usually an experienced sales consultant. However, since it is unrealistic to assume that the sales consultant will provide the definition of an optimal set of HLFs, our approach uses case-based reasoning in order to detect suboptimal indexing and to propose optimizing modifications of the used indices (see 3.2.3 for details).
Since the optimality does not solely depend on the sold objects, but also on the cognitive structure of the sales consultant and the customers, the expert has to judge each proposed modification. Moreover, he is in charge to give cognitively adequate names to the possibly modified HLFs.

# 3 System Description

The target of the sales consultation can be viewed as the determination of a possibly empty class $X_a \subseteq X$ of acceptable sales objects.
However, as already explained, we cannot presuppose that the customer is able to determine the class $X_a$ by his/her own. Hence, it is not sufficient to query for the desired value of each technical feature of the sales objects.

## 3.1 The representation of cases

The representation of the possible sales objects is frame based. The basic slots contain all possible technical features of the sales objects.
In addition to these slots the frame contains slots for HLFs. These HLFs are represented as slots with associated sets of possible values. To each HLF is a *dependence set* associated, which is a set of basic slots. For the dependence set of slots each value of the HLF defines default values. Hence by choosing a specific value for a HLF, one automatically chooses a set of default values for the basic slots; i.e. a set of technical features. See figure 1.

### 3.1.1 The formal description of the sales cases

We consider the following representation of sales objects. The representation of the cases for our case-based reasoning approach is grounded on the sales object representation:
Each object $x_i$ is characterized by a vector of attribute-value combinations:

$$x_i = \langle (a_1, v_{i_1}), (a_2, v_{i_2}), \ldots, (a_n, v_{i_n}) \rangle$$

Each attribute has a finite number of range values that can be chosen. The set of range values is denoted by $V(a_i)$. I.e.

$$\forall i \quad V(a_i) = \{v_1, \ldots, v_n\}.$$

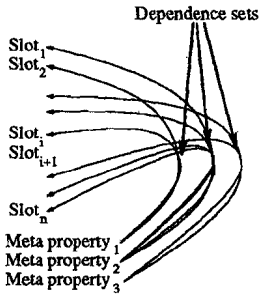



Figure 1: Schema of the frame representation of cases. Technical features and HLFs which are used for indexing the cases in multiple dimensions.

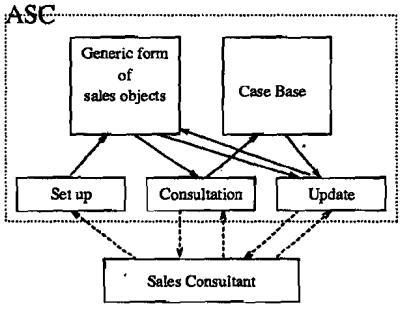



Figure 2: The system architecture of ASC. The arrows depict the major flow of information.

Then, the complete set of possible objects $X$ is given by:

$$X = \{x_i | \forall 1 \leq j \leq n)\ v_{i_j} \in V(a_j)\}.$$

### 3.1.2 High level features (HLFs)

A HLF is formally the same as a property, i.e. an attribute with a set of possible values. To each HLF $P_i$ is a dependence set of attributes $D(P_i)$ associated. The set of values $V_{P_i} = \{v_1, \ldots, v_n\}$ of $P_i$ determines default values for the attributes in the dependence set of $P_i$.
I.e. there is a mapping $Att : M_P \rightarrow 2^A$ which determines the dependence set of attributes of each HLF. The mapping $Def_{P_i}$ determines the default values for the attributes in $Att(P_i)$ for each possible value of $P_i$. I.e. $\forall V(P_i)$ there is a specified mapping

$$Def_{P_i} : V(P_i) \rightarrow V(a_{1,P_i}) \times \ldots \times V(a_{|Att(P_i)|,P_i}), where\ \{a_{1,P_i}, \ldots, a_{|Att(P_i)|,P_i}\} = Att(P_i).$$

One should note, that it is allowed to have nondisjoint sets of attributes associated to different HLF. This can be useful, if different high level features of the customer's needs respectively of the sales objects affect the same detailed sales object features. However, this nondisjointness may also cause conflicts between the default values assigned to attributes in the intersection of two HLF dependence sets. The resolution of such conflicts is explained later on.

## 3.2 Architecture and Operation of ASC - Automated Sales Consultation system

Figure 2 shows the principal architecture of ASC. ASC consists basically of two modules, *Set of Possible Sales Objects* and the *Case Base*, which have just a retrieval function within the system. Three further modules, *Set up, Consultation* and *Update*, are responsible for the different modes of operation of ASC.

### 3.2.1 The set up mode of ASC

All basic slots of the case representation are specified and named together with the range of admissible slot values. Furthermore, all initial HLFs are defined, named and their respective dependence set of basic slots are specified. The default value sets for the allowed HLF values are determined as well.

### 3.2.2 The consultation mode

The consultation mode of ASC is conceived to support the sales person in asking the right questions at the right time. Thus, ASC poses questions which should be answered by the customer after discussing it with the sales person. ASC tries to minimize the number of questions which are necessary in order to guarantee an appropriate choice of the sales object.
For that purpose, ASC starts with querying for the value of the first HLF among the HLFs of highest degree. Consecutively, ASC proceeds by querying for the yet unknown values of all HLFs of the highest level. As long as no conflicts among the assigned default values for the (basic) slots appear, it is up to the sales person to determine, *when* the assigned default values to the slots of the respective dependence set are verified. Verification, here, means that the assigned values are shown to the customer for either acknowledging or modifying the assigned values.
After each modification of an assigned slot value, ASC checks whether the currently assigned slot value combination of the dependence set of all HLFs becomes more similar to another HLF value. If so, this is indicated to the user and a change of the HLF value is performed together with the replacement of the default values for the slots of the dependence set, which have not yet been acknowledged or modified. This procedure continues until all basic slot values have been assigned and acknowledged or modified explicitly.

### 3.2.3 The update mode

The update mode contains the case-based reasoner of ASC. It is invoked after a successful sales consultation. Since it requires a lot of computational effort, it would thus preferably be run in the batch mode over night.
The case-based reasoner is used in order to determine - from prior consultation sessions - an adequate query strategy for the sales consultation process.
The query strategy can be modified in the following ways:

a) **Modifying the default values for the dependence set associated to a value of a HLF:** Assume the number of cases, in which a set of default values associated to a HLF-value - chosen by the customer - has been modified. If it is significantly greater than the number of cases where the complete default value set has been acknowledged, then this dominating modification of the default values is chosen as the new set of default values associated to the respective HLF value.

b) **Extending the set of values of a HLF/the creation of a new default value combination for the dependence set:** Assume the number of cases, in which the set of default values for the dependence set of a HLF is modified towards the same resulting slot value set. If it is greater than a prespecified threshold, then this dominating modification of the default values is chosen as a proposal for introducing a new HLF value, which has as its associated default values the above mentioned values.

c) **Modifying the dependence sets of the currently defined HLFs:** If particular slot values are often modified after they have been set to a default value due to the choice of HLF set of the respective HLF, then the inclusion of slots into a dependence set works as follows: If the finally chosen values of a slot $s$, in known cases, correlate significantly stronger with the values of a new HLF $P_n$ than with the old HLF $P_o$, in whose dependence set $s$ is contained, then ASC proposes to include $s$ into the new HLF $P_n$.

d) **Extending the set of HLFs:** ASC suggests the creation of new HLFs if a large number of the same slots is modified in a significant fraction of consultation sessions. Here, ASC lists the respective slots and proposes to create a new HLF covering the listed slots.

All modifications of the HLF structure are subject to expert's (in our case the sales consultant) confirmation. New HLFs or new HLF values must be properly named by the sales consultant, e.g. *security of transport is very important*, or *minimal price is requested.* This is to make the meaning of the HLF values cognitively accessible for the sales consultant as well as for the customer who has finally to decide among different values.

# 4 Conclusions

A new technique of case-based reasoning has been described, which makes possible to use CBR in areas where only very few cases are available. Multidimensional indexing as well as composing a solution from

multiple cases have been proposed. The possible modification of the indexing structure by modifying the dependence set of a HLF allows the implicitly used similarity measure to be asymmetric which appears to be psychologically much more plausible than symmetric similarity measures. So far, only symmetric similarity measures have been used in CBR. Moreover, the cognitive structure of sales consultation has been analyzed insofar, that the sales consultant can be strongly supported by an automatic sales consultation system.

# References


[1] A. Aamodt. Knowledge-intensive case-based reasoning and sustained learning. In *Current Trends in Knowledge Acquisition*, pages 1–20. IOS, Netherlands, 1990.

[2] R. Barletta and S. Mott. Techniques for employing case-based reasoning in automated customer service help desks. In *Proceedings Workshop on Artificial Intelligence for Customer Service and Support*, 1992.

[3] J. P. Callan, T. E. Fawcett, and E. L. Rissland. Cabot: An adaptive approach to case-based search. In *Proceedings IJCAI'91*. Morgan Kaufmann, 1991.

[4] K. D. De Jong and A. C. Schultz. Using experience-based learning in game playing. In *Proceedings of the fifth International Conference on Machine Learning*, pages 284–290. Morgan Kaufmann, 1988.

[5] K. Hammond. Learning to anticipate and avoid planning problems through the explanation of failures. In *Proceedings of the Fifth National Conference on Artificial Intelligence*, pages 556–560. Morgan Kaufmann, 1986.

[6] A. G. Hoffmann and S. Thakar. Acquiring knowledge by efficient query learning. In *Proceedings of the* $12^{th}$ *International Joint Conference on Artificial Intelligence*, pages 783–788, 1991.

[7] J. Kolodner and R. L. Simpson jr. A case for case-based reasoning. In *Proceedings of sixth Annual Conference of the Cognitive Science Society.* Lawrence Erlbaum Associates, 1984.

[8] J. L. Kolodner. Improving human decision making through case-based decision aiding. *AI Magazine*, (Summer), 1991.

[9] M. Redmond. Distributed cases for case-based reasoning: Facilitating use of multiple cases. In *Proceedings AAAI'90.* Morgan Kaufmann, 1990.

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

[11] S. Thakar and W. Tank. Knowledge Acquisition Phase Model for Customer Requirements Analysis. In *Proceedings of 2nd German National Workshop on Knowledge Engineering.* Berlin, 1990.

# *A priori* Selection of Mesh Densities for Adaptive Finite Element Analysis, using a Case Based Reasoning Approach.

NEIL HURLEY
Hitachi Dublin Laboratory,
O'Reilly Institute, Trinity College, Dublin 2, Ireland.
Email : nhurley@hdl.ie

**Abstract.** This paper describes the application of case based reasoning techniques to a complex domain, namely, mesh specification for finite element analysis. The case base provides a high-level store of information extracted through CPU-intensive numerical error analysis of previously solved problems, making it available for mesh specification before the simulation of new similar problems. Using this information, a near-to-optimum mesh is specified as input to the simulation engine, avoiding time-consuming computation during simulation. The paper describes the system, case representation, organisation and retrieval, and compares the CBR approach with the more usual rule-based approaches to this application domain.

## 1 Background

Finite element analysis (FEA) [7] is a powerful tool for solving engineering problems described by differential equations. In FEA, the continuous physical characteristics (for example temperature, pressure, fluid flow) of interest to the engineer, are approximated by a discrete model, in which a grid of mesh elements is generated across the geometrical domain and the numerical values for the physical characteristics are calculated at the grid points (usually referred to as the nodes). The values within the elements are approximated by piecewise continuous 'interpolation' functions. A typical problem from heat analysis is shown in Figure 1. (This problem will be used to explain our technique, and will be referred to from now on as Problem 1.)

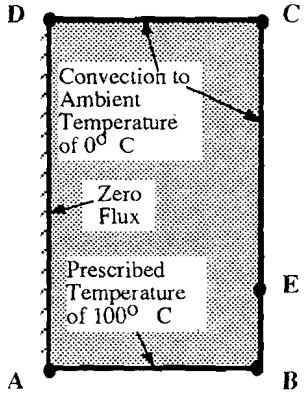



Figure 1 : Heat Conduction over a Flat Plate

The accuracy of the finite element model is highly dependent on the mesh. The greater the number of nodes used, the greater the accuracy. However, as the number of nodes is increased, the time required to complete the simulation also increases. There is therefore a trade-off between accuracy and efficiency and it is very important to find an optimum between the two.

Specifying appropriate densities for the mesh so that solution features are properly captured, is one of the most difficult tasks facing users of FEA. This issue is illustrated in Problem 1, in which the linear steady-state 2-dimensional heat conduction problem has a severe discontinuity in the boundary conditions in the lower right-hand corner (labelled by B in the diagram). The temperature profile shows a steep gradient due to this discontinuity between the points E and B. This will be correctly modelled only if a fine mesh density is used close to the corner. A much coarser mesh density will suffice further away from the corner (see Figure 2). Note that *a priori* knowledge of the temperature profile is required in order to correctly specify the mesh. Since, for most complex problems, time and memory limitations preclude the placement of a fine mesh over the whole domain, it is desirable to find an optimum mesh, with the mesh density throughout the domain varying according to local requirements. However, for many practical problems, it is not immediately obvious to the engineer or analyst where a fine mesh may be required.

The numerical approach to this problem is called *adaptive* finite element analysis. Essentially, the strategy is to solve the finite element problem a number of times, each time improving the quality of the mesh, until a satisfactory solution is found. An error estimator applied after each simulation locates the areas of high error and the mesh is refined in those areas. The process is repeated until a satisfactory error tolerance is obtained.

While the basic adaptive strategy is to start with a coarse uniform mesh throughout the domain, leaving the task of locating local phenomena which require mesh refinement completely to the error estimator, it has been

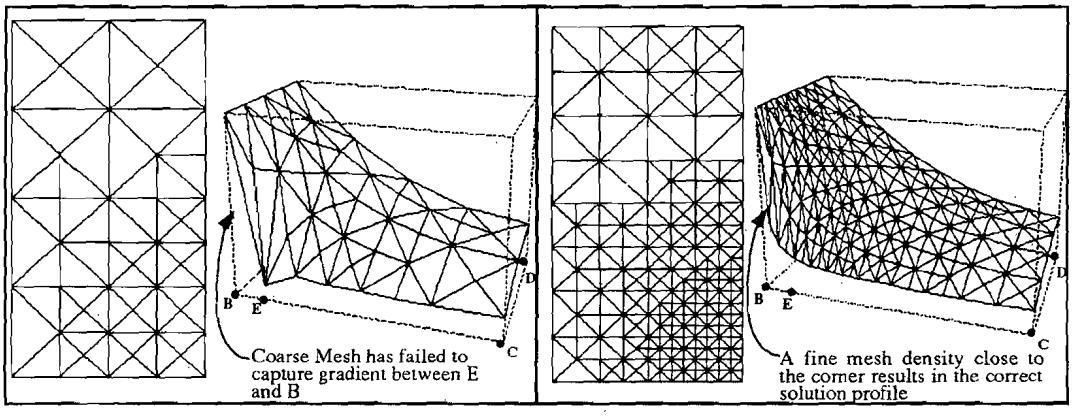



Figure 2 : Modelling the Temperature Distribution

observed (e.g. in [5]) that the effectiveness of adaptive strategies can be dependent on the initial mesh. In any case, it is desirable to avoid many iterations during the adaptive process, and one way to achieve this is to use *a priori* knowledge to set a near-to-optimum initial mesh. In practise, *a priori* techniques tend to be heuristic in nature, with the engineer or analyst relying on past experience to determine where a fine mesh will be required for a given problem.

In this paper, the application of case based reasoning techniques to the task of setting an initial mesh for an adaptive finite element simulator is described. Given a differential equation problem to solve, the system forms a solution strategy by accessing a case base of previously solved problems and matching the current problem with similar solved problems. The case base serves to augment *a priori* knowledge of a given problem by making available knowledge gained through *a posteriori* error analysis of previously solved, similar problems.

## 2 System Overview

Briefly, a case base of differential equation problems is stored, with each case containing a set of problem features and a set of solution profile features, extracted from the numerical solution. Solution features include any qualitative characteristics, for example, high gradient, which can be extracted from the numerical results data, (and which can only be modelled properly through the use of local mesh refinement). A frame representation of the problem (target case) is formed, and the problem features in this representation are matched against the problem features of the base cases. From the retrieved cases, a set of predicted solution features for the target case is formed. Meshing strategy routines generate the initial mesh given the predicted solution features. The adaptive finite element simulator solves the problem, refining the mesh, if necessary. The problem (with its solution) is then added to the case base.

## 3 Related Work

Much interest has focused in recent years on the application of knowledge-based techniques to the creation of problem solving environments for engineering and mathematical analysis. However, most of this work has considered rule-based approaches only. Within this body of work, rule-based mesh generation and adaptation systems have been considered. Expert systems to aid the geometrical problem of mesh generation (i.e. to ensure that elements are well-shaped) include the EZGrid system [1]. Tackling the issue of mesh adaptation, an expert-system for deciding when and how to refine or coarsen a mesh is described in [2]; Rank and Babuska [5] propose an expert system approach for selection of mesh adaptation strategies; and a blackboard architecture expert system, which makes use of boundary conditions and loading information to design a mesh refined at critical points, is described in [3]. Also of note is the work by Macedo et al. [4] who propose a knowledge-based approach to the selection of error indicators for mesh refinement schemes, based on a case analysis of several characteristic problems.

The case-based approach adopted in this research is motivated by the desire to create a flexible system which can augment its knowledge-base as more knowledge becomes available. In fact the importance of accumulating experience in knowledge-based engineering design/analysis systems has already been noted (e.g. in [6]). While a rule-based approach can provide a good coverage of well-understood problems, it will fail when new problems outside this coverage are presented. A rule-base which predicts the behaviour of simulation problems needs to take into account not only all the many features that these problems may exhibit, but also how these features interact. This is particularly difficult, since the behaviour when two features are present may be completely different to the behaviour when either one is present without the other.

It is felt that the case-based approach can propose solutions even when only partial knowledge is available, since it forms strategies based on similarity, without requiring this similarity to be grounded in explicit domain knowledge. Furthermore, the adaptive simulation engine at the back-end of the system can act as a teacher,

correcting the initial mesh design proposed by the front-end, and providing the means by which case indices can be improved to avoid the same pitfalls in later problem-solving episodes.

## 4 Case Representation and Retrieval

The solution profile from a finite element simulation is influenced by three categories of features, namely *equation features*, *geometry features*, and *boundary condition features*. By equation features, we mean equation types such as *parabolic*, *hyperbolic* or *elliptic*,[1] as well as characteristics such as non-linearity, size of coefficients, inclusion of sources, etc. Boundary condition features include the type of boundary condition (e.g. *insulated* or *fixed flux*) as well as size of coefficients, etc. Geometrical features include cracks, corners, protrusions, obstacles, etc. Since there is a large body of differential equations, describing many different behaviours, it is necessary to focus on some sub-class of problems. We are applying our techniques to steady-state diffusion and advection heat transfer problems, that is, the class of problems described by the following partial differential equation :

$$-\nabla.(k(x,y)\nabla\varphi) + v.\nabla\varphi = q(x,y) \quad (1)$$

In the context of heat transfer, $\varphi$ represents the temperature profile, $k(x,y)$ the material conductivity, $v$ the (fixed) flow field, and $q(x,y)$, the heat source or sink. Associated with this problem, there are four categories of boundary condition, namely,

*Fixed Temperature* : The temperature is prescribed on the boundary,
*Fixed Flux* : A fixed heat flux is maintained across the boundary,
*Insulated* : No heat transfer across the boundary,
*Convection* : Convection to the ambient temperature.

Work is focusing on determining mesh densities for different combinations of these equation and boundary condition features.

Each case contains a full problem description, consisting of the equation to be solved, the domain over which it is solved and the boundary conditions on each boundary. Such a description is shown in Figure 3.

```
(pde-problem
   (geometry     (a        (point (0 0)))
                 ...
                 (ab       (line a b))
                 ...
                 (r1 (polygon ab bc cd da)))
   (regions      r1)
   (boundaries   ab bc cd da)
   (variables    (TT))
   (equation1    (TT)
                 (= (diffuse TT) 0)
                 (bcond1 (= TT 100) on ab)
                 ...
                 (bcond4 (= (ngrad TT) 0) on da))))
```

Figure 3 : Problem Description

To devise a scheme for matching target and base cases, we note the following:

(i) Relationships between case features are important. This is illustrated by Problem 1. The high gradient profile at corner B results not because of a single feature; rather it is due to the *relative location* of the Convection and Fixed Temperature boundary conditions.

(ii) A qualitative representation of the problem does not suffice. It is important to establish the *significance* of features on the overall solution profile, and this can only be accomplished if the strength of the feature's effect is taken into account.

Comparing relative strengths of features between different problems requires that the problems be normalised in some manner. This may be achieved by calculating a characteristic length of the domain, and using engineering approximations to estimate the average temperature and heat flux on the domain, based on the initial data. Of interest from a meshing point of view are those local areas of the domain where the heat flux is much greater than the average.

To facilitate matching, a taxonomy of problem features and feature relationships is maintained (see Figure 4). The case indices consist of specific relationships between the problem features coupled with specific solution behaviours. For example, the case corresponding to Problem 1, would be indexed by the predicate

```
Connected-at(bcond1, bcond2, B)
```

where `bcond1` and `bcond2` are the two boundary conditions which meet at the point `B`, coupled with the solution predicate

```
Close-to(soln-feature1, B)
```

[1]This terminology refers to the order of the various spatial and time derivatives in the equations.

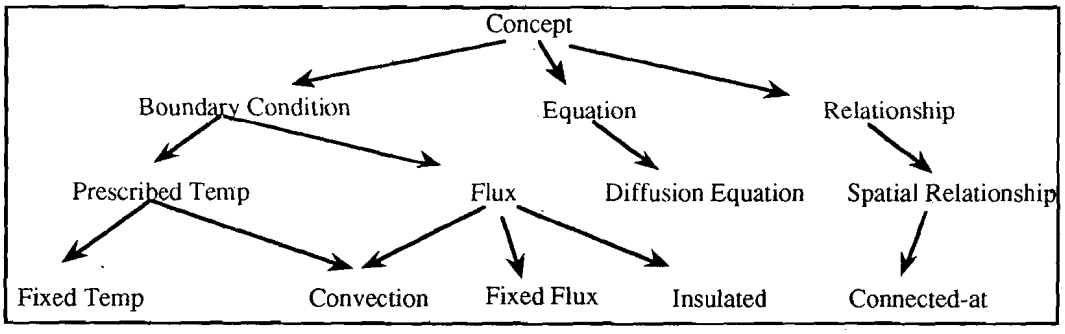



Figure 4: Part of the Taxonomic Hierarchy

where `soln-feature1` is the solution feature corresponding to the high gradient at the point B. Associated with each solution feature, the case holds a mesh density parameter which indicates how much the mesh needs to be refined in order to capture the feature properly. In essence, the index corresponds to a rule which is true for the specific case to which it is attached. Complex cases may contain many such indices if more than one solution feature appears in the solution profile.

Case retrieval proceeds by firstly extracting all the relationships which hold in the target case. Cases are stored in a hierarchy which classifies them according to the equation type e.g. `linear-diffusion-case` or `diffusion-advection-case` etc., and the type of problem features which occur in their indices e.g. `fixed-temp-case`, or `convection-case`. Matching is performed only against those cases which are concerned with the same equation type and whose indices contain problems features which also appear in the target case. Partial matching may be achieved by generalisation and specialisation of the index components along their taxonomic hierarchies. Many cases may be retrieved for one given target case, corresponding to matches against different target relationships. If more than one case matches a particular target relationship, then quantitative information is taken into account. For the above example, the numerical values of the boundary conditions are examined, and the case for which the numerical values agree most closely is preferred.

## 5 Simulation and Learning

Once the mesh design is processed by the mesh generator, the problem is simulated and the mesh is adapted using numerical techniques. It is possible that the problem may exhibit behaviours which were not predicted by the case-based pre-processor, in which case a new index to account for this behaviour should be generated. Methods for accomplishing this task still need to be investigated, but at least this can be done through user consultation. The other possible error is that predicted solution features are not actually present in the actual solution profile. This requires not only that the index in the original target case, but also that the index in the base case with which it was matched be modified. One method is to search for specialisations of the base case index which do not hold for the target case. For example, in Problem 1, the high gradient feature results not just because the boundaries are connected, but, more exactly, because they are *incompatible* i.e. there is a discontinuity at the corner. The specialisation to the `incompatible` relationship will be necessary when, in a new problem solving episode, a target with the same boundary condition features is found not to exhibit the high gradient feature in its solution profile. By adjusting its indices in this manner, the system can learn to avoid mistakes it has made in the past.

## 6 Conclusions

A case-based reasoning system for mesh design of finite element problems has been described. The system is currently under implementation . It is believed that this approach has certain important advantages over more traditional rule-based approaches to this domain, in particular, its ability to deal with problems where a complete *a priori* understanding of possible behaviours is not possible, and its ability to learn from past problem-solving episodes.

## References

1. A.E. Andrews:*Progress and Challenges in the Application of Artificial Intelligence to Computational Fluid Dynamics*,, AIAA Journal, vol. 26, no. 1 (1988)
2. J.F. Dannenhoffer & J.R. Baron: *A hybrid expert system for complex CFD problems*. AIAA Paper 87-1111, p. 99. Proc. AIAA 8th Comput. Fluid Dyn. Conf., Honolulu (1987).
3. E. Kang & K. Haghighi: *A knowledge-based* a-priori *approach to Mesh Generation in Thermal Problems*, International Journal for Numerical Methods in Engineering 35, 915-937 (1992)
4. C.G. Macedo,Jr., J.C. Diaz & R.E. Ewing: *A Knowledge-based System for the Determination of Activity Indicators for Self-Adaptive Grid Methods* in : E.N. Houstis et al, (eds.), Intelligent Mathematical Software Systems 133-141 North Holland (1990)

5. E. Rank & I. Babuska: *An Expert System for the Optimal Mesh Design in the hp-Version of the Finite Element Method*, International Journal for Numerical Methods in Engineering 24, 2087-2107 (1987)
6. W.W. Tworzydlo, J.T. Oden: *Towards an antomated environment in computational mechanics*, Computer Methods in Applied Mechanics and Engineering 104, 87-143 (1993)
7. O.C. Zienkiewicz & K. Morgan: *Finite Elements and Approximations*, Wiley & Sons, (1983)

# Integrating Semantic Structure and Technical Documentation in Case-Based Service Support Systems

Gerd Kamp
Computer Science Department,University of Hamburg
Bodenstedtstr.16,2000 Hamburg 50

**Abstract**

Help desk systems are one of the most successful application areas of case-based reasoning. However, case-based reasoning techniques cover only parts of the whole help desk scenario. One missing part is providing access to the technical documentation. Combining these becomes especially important in the area of service support systems, where the service person has no access to the printed documentation. This paper presents a concept how to integrate CBR and technical documentation for service support systems.

## 1 Introduction

One of the most successful application areas of Case-Based Reasoning[1] is the domain of so-called 'Help Desk Systems'. 'Service Support Systems' essentially serve the same purpose but in a slightly different environment. In this paper we investigate extensions to 'classic' CBR that are needed for service support systems. Therefore we first explain how we want to understand the terms help desk system and service support system and then define some requirements for service support systems. In the remainder we describe a concept for a CBR based service support system that meets these requirements.

## 2 Help Desk Systems

With the development of the personal computer in the early 80s the need for supporting new kind of users, i. e. managers, technicians, secretaries, arose. Therefore many companies created Information Centers, to assist and control the use of PCs within the company.

The first systems used by the Information Centers were database management systems to help with the information about the clients hard- and software. With the advent of expert systems intelligent job aids for Information Centers could be developed. Thus expert system theorists called this systems *diagnostic expert systems*, training assistants called them *intelligent job aids*, and the people at TIs Information Center[2] called them *help desks*.

First used for computer-related problems, help desks today can refer to any computer-based system that aids people in providing assistance via phone. Users needing advice contact[3] a human operator or Customer Service Representative[4]. In a simple operation the CSR listens to the user describing the problem, and then provides a recommendation based on his experience.

Unfortunately, such operators are hard to find. Moreover, as equipment gets more complex, it's hard to find anyone to man a help desk who really understands everything a user might ask about. Most CSRs know how to deal with the standard, frequent questions and rely on manuals and notes to come up with a solution for harder, less frequent problems.

### 2.1 Requirements

Creating a help desk system therefore is the task to assist the CSR with retrieving and storing with the following kinds of information.

**Information about events and users** This means storing and retrieving records of user configurations, contacts, etc.

[1] we will abbreviate this as CBR in the following

[2] TI developed such systems, e.g. the Hotline Advisor for assisting customer support people in solving customers problems related to printers

[3] normally by phone

[4] abbreviated as CSR below

**Information about products and services** Information of that kind is provided with the technical documentation of the products, such as a manual describing the part structure or a diagnostic manual etc.

**Information about known problems** This is the information gained in past calls to the help desk and often stored as notes or protocols of the calls.

**Knowledge about how to solve problems** Knowledge about problem solutions[5] consists of procedural knowledge,i. e. how to proceed in a certain situation, heuristic knowledge or behavioral models.

Assisting the first kind of information is often done with conventional data-base techniques, where the second could be supported by information retrieval and online document retrieval. The third item is best assisted by case-based reasoning systems, whereas the fourth is due to rule-based or model-based diagnostic expert systems.

For that reason, most CBR systems used in the field of help desk systems provide the CSR with information about previous calls, replacing the paper-based notes and protocols. The other kinds of information are provided by other sorts of systems like databases or information retrieval systems.

# 3 Service Support Systems

A situation similar to help desk systems is given if one is to assist the work of a service man or technician on location. But beside the task of supporting the technician in finding the right diagnosis, a service support system has to serve him in some other parts of his work as well.

## 3.1 Additional Requirements

**Planning visits** Because of the different tasks of a technician and the steadily increasing palette of machines (and their variants) it is impossible for him to remember all installations he is responsible for. Regular maintenance of the machine including the determination of critical parameter values and the exchange of wearing components is often part of the contract for complex machines.

In order to plan a visit to a customer the technician has to know the details of the installation, i.e. to take the right components with him. A service support system has to provide the technician with this information. This corresponds to the information about users in a help desk system.

**Online Technical Documentation** It is impossible for the technician to carry the whole technical documentation for a large variety of machines. Thus a service support system has to provide the technical documentation as well as the experiences. Therefore including facilities to access the documentation is mandatory for a service support system.

**Protocols** To document his job the technician has to write reports of his visits. In contrast to pure help desk systems not only the failures are interesting but also the values of certain parameters over time, e.g. to fulfill some legal constraints. These protocols have to be stored by a service support system.

## 3.2 System Design

Another important point is the emphasis on *support* in service support system as opposed to automatic operation. This is motivated by the following two observations:

Due to their job, technicians are used to work alone. They are the ones to make decisions and to take the responsibility for it. Therefore, every kind of tutorship has negative impact.

Systems which make decisions, e. g. a diagnosis, by themselves, typically make faults. When these decisions are treated too offensively by the system, the technician will soon refute the system[6].

A service support system therefore has to leave the initiative to the technician, it serves as a system that provides the information the technician wants to have in a particular situation. This is in contrast to a model where the initiative belongs to the system and the user is to provide the information which the system cannot deduce.

[5] The distinction between information and knowledge about problems is a bit fuzzy

[6] This observations are general, but in the context of service support systems their impacts are crucial

# 4 A CBR Based Service Support System

## 4.1 Classic CBR approaches

There are several papers describing the state of the art in CBR[8, 4, 1]. We therefore only give a short description of a classic CBR approach in technical domains and especially help desk systems.

**Case Representation** In most systems cases are represented as attribute[7]-value vectors with some basic value types such as *numbers* (*intervals*), *strings* and *sets of symbols* describing nominal or ordinal types.

**Feature Similarity** The single attribute similarity is mostly obtained by equality testing. Some systems allow ranges and deviations for features involving numbers, and implement some spell checking routines and substring testing for string features.

**Similarity Measure** The similarity measure comparing the current situation with a stored case is often a function combining single attribute similarities to a value in the interval $[0 \ldots 1]$ (or $[-1 \ldots 1]$). This is often a weighted sum of the single attribute similarities, or a function based on the contrast rule by Tversky[9].

**Retrieval** In a first step a set of relevant cases is selected, often on a selection of those cases that contain mandatory features. Then cases are sorted according to a similarity measure in a second step, determining the nearest neighbors of the presented case. Following steps eventually involve the modification of the most similar case according to the given situation, but there are few implemented systems and especially in the domain of classification[8] this step is often not needed.

In this kind of CBR systems there is no way to represent the structure of a machine as well as the history of the features over the time. Additionally there are no means of integrating the technical documentation. In the following we will describe a concept for a system realizing these requirements.

## 4.2 Structured CBR – The AMS Appraoch

Normally cases are represented by flat feature-value vectors. But often, and especially in the field of help desk or service support systems, there is knowledge about the structure of the domain, i. e. about machines and plants to be supported.

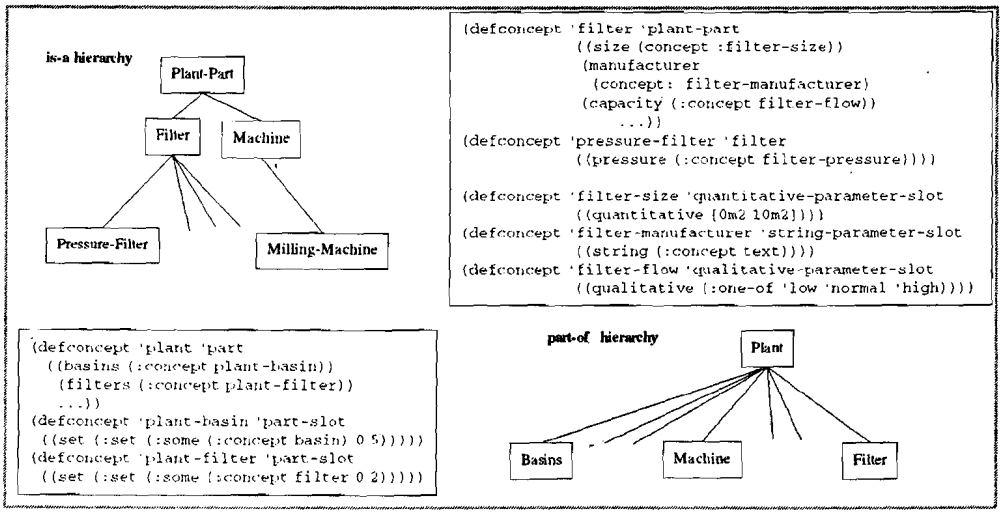



Figure 1: Parts of the domain structure of AMS

This knowledge enables us to model the structure of the domain via *is-a* and *part-of* relations as in frame knowledge representation systems like KL-ONE or KEE[3][9]. With this kind of structure we can for example represent the fact that a milling machine is a kind of a machining tool, or that a pressure-filter is a kind of filter. With the part-of relation we can describe that a manufacturing plant has, among other, some basins for the cutting fluid, some filters to separate chips and dirt from the cutting fluid, some machining tools etc. (see Figure 1).

[7] or feature, parameter

[8] i.e. finding a diagnosis

[9] KL-ONE and KEE are chosen among the variety of frame representation systems to illustrate the design space of frame representation systems

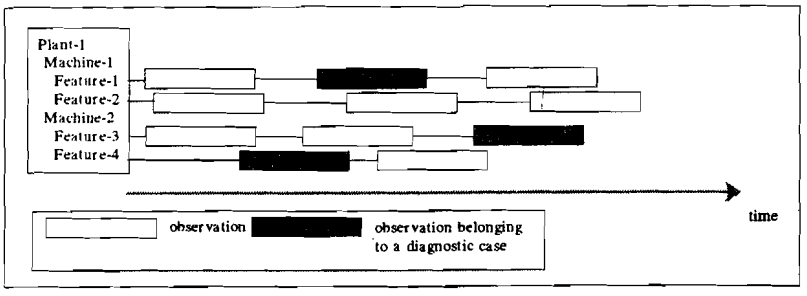



Figure 2: A Schematic Plant History

Recent CBR projects like AMS[10][2] and INRECA[5] therefore use frame-representation languages to structure the domain.

**Domain Structuring** The domain is structured via the above mentioned *is-a* and *part-of* relations. One defines *concepts* representing domain objects and states relations between them.

For example one might define a concept `filter` as a kind of `plant-part` with the measurable parameters `size`, `grammes-per-square-meter`, `capacity` etc. It is a leaf node in the part-of hierarchy and contains no part slots. Special filters as a `pressure-filter` would then be defined as subconcepts of filter e. g. by adding a parameter-slot `pressure`.

Measurable parameters are defined as subconcepts of the class `parameter-slot`. They can contain nominal, qualitative (ordinal) and quantitative values as well as texts[11]. `part-slots` are relations with a range that is a subconcept of `part`. They might have number restrictions as known by KL-ONE.

**Case Representation** As stated in section 3, a service support system has to store and retrieve the normal values of some features as well as the feature values determined during the diagnostic process.

**Machine and Plant Histories** In order to store feature values of a certain machine the technician has first to enter the structure of the machine, i.e. he has to instantiate the domain concepts in order to get concrete *instances*. The slots of these instances are then filled by the feature values.

When a new feature value is entered, it corresponds to an *observation* made by the service man. An observation is a quadruple $(object, slot, value, time)$, so that the slot value of a feature is a list of pairs $(feature, time)$, representing the history of this feature. The *plant history* is then the collection of the*feature histories*.

**Diagnostic cases** Diagnostic cases are represented as $(reference, characterization, situation, diagnosis)$ where $reference$ is a pointer to the plant the failured occured on, characterization is a (short) textual description of the failure, whereas *situation* is a set of previously made *observations*, and *diagnosis* is a list of *diagnosis-steps*. *diagnosis-steps* themselves are triples $(hypothesis, test, result)$ with *hypothesis* as a hypothetical observation, test an evaluation method and *result* an observation confirming or refusing the hypothesis.

**Feature Similarity** In order to allow a rather broad scope of queries, including exact matches, we define a set of comparing relations for each basic type $A$. The relations are themselve partially ordered in the sense of set inclusion. The minimal element of this ordering is the diagonal relation $id?(a, b)$ consisting only of the pairs $\{(a, a)|a \in A\}$, whereas the maximal relation is the all relation $all(a, b) = A^2$.[12]

For some types one then can define other relations as $is?(a, b)$ meaning set inclusion for sets of symbols or range inclusion for intervals. For strings $is?(a, b)$ could be interpreted as substring occurrence. Another step in this relation hierarchy could be $sect?(a, b)$ meaning a non-empty section between two sets or ranges. Furthermore $near?(a, b)$ could be defined, specifying that $a$ is not too far apart from $b$ in the sense of a distance measure based on the type the difference in the case of numbers, or something like a hamming distance[13] in the case of strings. The different stages of feature relations provide a first way to generalize from a given situation in order to perform a similarity search.

**Instance Similarity** Another dimension of generalization is provided by the domain structure, in that we can generalize within the *is-a* or *part-of* hierarchy. The approach is best described by the rough

[10] AMS is a case-based support system developed by the author at University of Hamburg's Artificial Intelligence Laboratory in cooperation with a manufacturer of cooling lubricants in order to aid it's technical staff.

[11] in AMS dimensions can be defined and used in quantitative Slots

[12] Because some of the comparing relations are no equivalence relations (e. g. they are not transitive) the relations could not be embedded in the lattice of the equivalence relations over $A$.

[13] i.e. allowing a certain number of insertions, deletions and wrong characters when searching the substring $a$ in the string $b$. This would implement a kind of spell checking function

definition that 'similarity is equality on a more abstract (or general) level' and corresponds to the set-theoretic semantic of concepts in KL-ONE[3]. This can overcome the limitation of flat feature vectors when determining for example, that `Relais-1` and `Relais-2` are syntactically different attributes, but have semantically the same function in functionally and structural identical subparts of a machine[6].

The specification of a `pressure-filter` in a certain query can then be replaced by a `filter` allowing to match all other kinds of filters. This could be accomplished by defining relations similar to the ones presented in the previous section over the set of all concepts. The relations between instances are then defined by the relations of their corresponding concepts.

For example, the diagonal relation $id?(a, b)$ is fulfilled if the instances $a$ and $b$ are instances of the same concept, whereas $is?(a, b)$ would have the semantics that the set of subconcepts[14] of $a$ is a subset of the set of subconcepts of $b$, or to say it in another way, if $a$ is a subconcept of $b$. In a similar way, $near?(a, b)$ maybe defined as the path length between two concepts in the hierarchy.

**Retrieval** Retrieval is performed by formulaling *queries*, i. e. conjunctions or disjunctions of *patterns* of observations, hypotheses etc., resulting in sets of machine histories or diagnostic cases that contain observations etc. that match this query.

One of the requirements made in section 3 was a user centered system design. This includes the specification of the similarity measure used in order to answer a certain query. The user can specify different comparing relations on the feature as well as on the instance level, where the default similarity measure used is testing the various parts of the observation via the $is?(a, b)$ relation resulting in a kind of subsumption test.

Thus the retrieval cycle is as follows:

1. **Formulate** In a specific situation, the user formulates a query and retrieves the items matching this query.
2. **Inspect** He/she inspects some of the retrieved cases. If he gets too many or too few matches he may reformulate the query, specializing or generalizing it, respectively.
3. **Adapt** If he finds an interesting match, he adopts this match to the current situation and proceeds.

# 5 Incorporating the Technical Documentation

The concept described in the last section solves the tasks of storing and to retrieving the structure of machines as well as their history and diagnostic cases associated with them. It does not yet provide any means to incorporate the technical documentation.

**Additional Basic Types** The main idea for incorporating the technical documentation is to broaden the range of basic feature types. Adding types used in hypermedia systems, such as *sound*, *pictures* and *video* allows us to store the documentation. Because the structure of the documentation is similar to that of the defined domain structure[15], moreover, the domain structure often is derived and acquired from the documentation, it is easy to incorporate the documentation into this structure, for example by providing the relevant parts of a components manual as additional slots of the concept describing that component.

**Structure as Hypertext** Having incorporated the technical documentation into the concept taxonomy and partonomy, one can use the *is-a* and *part-of* hierarchies as a link structure similar to a hypertext system, allowing the user to easily navigate through the so constructed online manual.

The new basic types can not only be used to incorporate the documentation, they are also useful in modeling the domain itself.

**Nominal and ordinal parameters** There are a lot of situations where it is much easier for a user to have sounds or pictures describing ordinal or nominal value types than the normally used symbol sets. For example, one can use sound to illustrate different noises of a part representing correct and false behavior. Another example are dip slides that are used to measure the pH-value or pictures of bacteriological cultures to determine a germination index.

**Visualization** Videos and Pictures could be used to visualize a certain test, e. g. how to replace a defect part.

# 6 CBR and Information Retrieval

Closely related to CBR is the field of Information Retrieval[7], which is mainly the task of retrieving a set of documents similar to a list of keywords (or another document). This is often accomplished by the

[14] including the concept

[15] i.e. the structure of machine manuals (chapter, sections etc.) correspond to the part structure of the machine

use of *indexed texts*.They are an alternative to the *string* type when larger texts need to be stored and retrieved. They allow the efficient retrieval of similar texts, and the method of relevance feedback[7] used to improve a query in information retrieval fits into the main retrieval loop mentioned in 4.2. Moreover many information retrieval systems also have different retrieval modes ranging from boolean retrieval to complex similarity functions similar to the relation hierarchy in 4.2 and 4.2.

Indexed texts than can assist in overcoming a problem of CBR projects, the problem of inadequate descriptions of old cases. When starting a CBR project one is often told that there are lots of cases already acquired. But when it comes to the point of getting the cases there are fewer than previously said, and they are in the wrong format, mostly protocols.

Protocols have to be converted into the more structured form of cases, which is a time consuming process. With indexed texts, protocols can be used directly for retrieval. For example the *characterization* part of a diagnostic case could be an indexed text. Then old cases, or cases the service man has no possibility or time to enter the formal description, could be first given in an textual description (and turned into a formal one later). Indexed texts therefore provide a migration path from unstructured to structured representation of cases and should be added as another basic type.

# 7 Summary & Outlook

In this paper we introduced service support systems as user-centered systems related to help desk systems and presented a concept for realizing them. This concept differs from other CBR systems in that it uses knowledge about the domain to structure the cases. Additionally it supports two types of cases: machine histories and diagnostic cases. The retrieval methods are based on a semantic similarity measure different from the more syntactic measures in classic CBR systems. The user itself can modify the similarity measure for a certain question to broaden or confine the retrieval.

Adding indexed texts, sound, pictures, and video as basic types allows us to integrate the technical documentation,which is often missing in normal CBR based help desk systems. They also allow to describe the cases in a more natural way and facilitate the process of converting existing paper-based case descriptions.

Nothing has been said about case adaption and learning. How this can be done using the classification and recognition capabilities of KL-ONE is part of current research.

# References


[1] K.D. Althoff, S. Wess, B. Bartsch-Spörl, D. Janetzko, Frank Maurer, and Angi Voss. Fallbasiertes Schliessen in Expertensystemen: Welche Rolle spielen Fälle für wissensbasierte Systeme. *KI*, (4):14–21, 1992.

[2] Gerd Kamp. Ähnlichkeit in AMS. In *Workshop: Ähnlichkeit von Fällen beim fallbasierten Schliessen*, pages 83–86. K.D. Althoff and S. Wess and B. Bartsch-Spörl and D. Janetzko, 1992.

[3] Peter D. Karp. The design space of frame knowledge representation systems. SRI AI Center Technical Note 520, SRI International, 1993.

[4] Janet L. Kolodner. Improving human decision making through case-based decision aiding. *AI Magazine*, 12(2):52–68, 1991.

[5] M. Manago, R. Bergmann, N. Conruyt, R. Traphöner, J.Pasley, J. LeRenard, F. Maurer, S. Wess, K.D. Althoff, and S. Dumont. Casuel: A common case representation language. Technical report, INRECA, 1993.

[6] G. Pews, F. Weiler, and S. Wess. Bestimmung der Ähnlichkeit in der fallbasierten Diagnose mit simulationsfähigen Maschinenmodellen. In *Workshop: Ähnlichkeit von Fällen beim fallbasierten Schliessen.* K.D. Althoff and S. Wess and B. Bartsch-Spörl and D. Janetzko, 1992.

[7] Peter Schäuble. A tutorial on information retrieval. In *Proc. of the 1993 Workshop on CBR.* AK CBR of the GI, 1993. (to appear).

[8] Stephen Slade. Case-based reasoning: A research paradigm. *AI Magazine*, 12(1):42–55, 1991.

[9] S. Wess. PATDEX – ein Ansatz zur wissensbasierten und inkrementellen Verbesserung von Ähnlichkeitsbewertungen in der fallbasierten Diagnostik. In *Expertensysteme 93*, pages 125–138. F. Puppe and A.Günter, 1993.

# CABATA — A hybrid CBR system

Mario Lenz
Department of Computer Science
Humboldt-University
Berlin

**Abstract**

This paper presents CABATA, a hybrid case-based reasoning system that has been developed at the Department of Computer Science, Humboldt-University, Berlin. The most characteristic feature of the system is the combination of model-based and case-based reasoning within a hybrid architecture.

## 1 Introduction

Within the framework of CBR research at the Department of Computer Science at Humboldt-University, Berlin, the CABATA-system has been — and is still being — developed. The system was designed to pay particular attention to the combination of domain-specific knowledge and classical CBR methods within a hybrid architecture. The cooperation of both, the rule-based and the case-based reasoning strategy, is expected to show significant improvements concerning all phases of CBR:

- efficient case retrieval (using indexing),
- the matching of cases,
- storage of cases and meory organization,
- learning beyond the scope of CBR.

The key features of the CABATA-system are

- integration of user-defined rules guiding the inference within the CBR-like inference engine
- dynamic similarity assessment of feature values via user-given context graphs
- incremental classification by subsequently modifying similarity knowledge
- EBL-like acquisition of domain knowledge.

The outline of the paper is as follows: Section 2 gives a short introduction to the domain chosen to serve as an example to demonstrate the hybrid architecture. Section 3 describes the way domain-specific rules can be defined and are used within the CABATA-system. Section 4 sketches the usage of context graphs to assess the similarity between symbolic feature values. Section 5 shows two ways efficient indexing can be implemented, section 6 describes a first attempt to implement learning strategies. Section 7 shortly discusses the CABATA-system in relation to important work in the literature (7.1) and lists some of the research topics (7.2) within the CABATA-system in the near future. During the whole paper a basic knowledge of case based reasoning methods is assumed.

The CABATA-system has been implemented using C++. Currently, CABATA runs on a PC 386/486 and requires Microsoft Windows©3.1.

## 2 The chosen domain

The simulation of a travel agency was chosen to serve as an example to demonstrate the CABATA[1] architecture. However, all parts of the inference engine have been implemented independently of the domain under consideration[2]. The task of CABATA is to choose one of its stored cases describing past holiday trips as a suggestion for a new trip satisfying user-specified conditions. These conditions are given as case descriptions and include features such as

[1] CABATA = CAse BAsed Travel Agency

[2] In fact, all parts of the whole program except for the graphical user interface.

- the type or aim of holiday (e.g. recreation, sporting activity, language course etc.),
- the chosen region (both general descriptions such as mountains or sea, and particular countries, cities or areas),
- the wanted means of transport (e.g. coach, car, train or plane),
- the season or month of the holiday trip,
- the maximal price,
- ...

While some features are numerical ones, others may take symbolic values. This will be of particular importance in section 4.

It seams important to point out here the difference to other CBR systems dealing mainly with technical diagnosis (e.g. [3, 4, 5, 6, 7, 17]):

1. In contrast to technical devices no causal relations guiding the inference are available.
2. The terminology describing the domain is not well defined: Such terms as "Recreation" are highly ambigous.
3. The diagnosis of a case itself is not an atomic object, such as a fault number or a descriptional string. Rather it is a structured object: A proposal of a new holiday trip similarly structured as other cases.

The process of selecting the best applicable case from memory is designed to be an incremental one, i.e. after a suggestion has been made by the system, the user may modify the set of conditions and the matching knowledge to adapt the solution.

# 3 Integration of domain-specific rules

Rules within the CABATA-system have the form

`IF <feature1> <rel1> <value1> THEN <feature2> <rel2> [<value2>].`

where

| | |
|---|---|
| `feature1` | contains the feature of the case description that — when satisfying the given condition — implies a certain modification of the inference. |
| `rel1` | describes the relation between `feature1` and `value1`. This may be one of the ordinary arithmetic relations ($=, \neq, <, >$ etc.), or a relation named `isA` to express if the value belongs to a certain type hierarchy, i.e. is a sub-type of `value1`. |
| `value1` | is the value to which `feature1` is compared using `rel1`. |
| `feature2` | contains the feature of the case description for which a certain condition must hold. |
| `rel2` | states the relation to `value2` thas must hold for `feature2`. Here two types are possible: Firstly, as above it may be one of the ordinary arithmetic relations or the `isA` relation. Secondly, it may be a statement indicating the importance of `feature2` when comparing two cases (reaching from `not important` to `most important`). In the latter case `value2` cannot be selected. |
| `value2` | gives the value to which `feature2` is compared using `rel2`. |

On the basis of this scheme, rules may be used in two different ways:

1. The dynamic similarity assessment may be influenced by increasing or decreasing the importance of particular features if certain conditions hold, e.g.

   `IF HOLIDAY_TYPE = CITY THEN SEASON IS OF LESS INTEREST.`
   stating that, when planning to visit a particular city, the season is less important than usually.

2. These rules may serve as restrictions (constraints) when demanding that a particular feature be of a certain value, for example

   `IF REGION isA SEA THEN REGION MUST NOT BELONG TO MOUNTAINS.`
   helps to avoid anomalies caused by the propagation of similarity values through the context graphs (see section 4) as:

   - The Alps have something in common with Italy[3].
   - Italy has something in common with the Mediterranean Sea.

[3] To `have something in common` expresses that the similarity value between the mentioned feature values is different from 0, the actual value is of no importance here.

- Hence: The Alps have something in common with the Mediterranean Sea.

Thus, in combination with the similarity graphs described in section 4 the integration of rules allows for a similarity assessment which depends on the context of the comparison of the attribute values.

# 4 Context graphs

For every symbolic feature being part of the case description a context graph is constructed containing information about both

- generalization and specialization
- similarities

between all possibile[4] attribute values.

These graphs have to be defined using binary relations between particular feature values. To serve as an aid during this process, a tool has been constructed — first of all to allow larger parts of the graphs to be displayed and edited. Thus the risk of local acceptable changes leading to global inconsistencies[5] is reduced (see the example above).

The similarity between two values of a feature itself is expressed using a number of measures ranging from `nothing in common` via `similar` to `identical` thus giving reasonable flexibility to describe domain knowledge.

At run time, that is when classification of a case is requested, this context graph is used to determine the similarity between different values of symbolic case attribute[6]. If the corresponding nodes of the two attribute values are connected, i.e. if a binary relation has been specified for these two values, a numerical equivalence for this relation is returned. If they are not directly connected the similarities are propagated through the graph. Thus the larger the distance between two nodes the weaker the similarity.

# 5 Indexing

To implement an efficient case retrieval, CABATA is designed to employ a twofold indexing strategy:

1. By applying a *passive* indexing to the case database stored cases are excluded from the classification if they do not satisfy certain conditions required by the case to be classified.
2. On the contrary, *active* indexing directly searches for cases with certain attribute values.

To give an example how both strategies work together, imagine that the current problem case expresses that somebody wants to go on a sporting activity holiday to Italy in winter:

- Passive indexing should exclude all cases dealing with city trips (wrong type of holiday), other destinations, taking place from April to September etc. However, a reference case describing a bathing holiday in October at the Mediterranean Sea could not definitely be excluded.
- Active indexing, on the other hand, could assume that the customer actually wants to go skiing. Thus cases describing holiday trips to the Italien Alps from October to March could be searched for.

So active indexing can be seen as a more sophisticated method using implicit assumptions not explicitely given in the description of the problem case. However, it is not yet clear how these implicit assumptions can be derived. Probably the attempt of learning *determination rules* (section 6.2) will give useful hints. Currently, only passive indexing is applied within CABATA.

# 6 Learning

One objective of CABATA was to investigate methods that could enable the system to learn beyond the mere storing of cases. Currently, two approaches are taken into account:

[4] Of course, during run time the set of all possible values of a feature may be modified.

[5] For instance, an acceptable modification of a single binary relation between two feature values may result in completely different values being similar to some extent. This is due to propagation of similarities through the graphs.

[6] Numerical values are compared by employing a sigmoid function.

### 6.1 Modified Explanation-Based Learning

Since pure *explanation based learning* ([10, 11, 12]) seams not applicable[7], a slightly modified strategy is applied: Similar to [13] the user is asked for matching knowledge when a diagnosis is finally selected for a case and discrepancies concerning that diagnosis are detected (i.e. a user-defined constraint is violated or there are other cases regarded significantly more similar).

### 6.2 Modified Learning from Examples

In a later stage, a knowledge acquisition based on the Machine Learning paradigm of *learning from examples* ([1, 2, 16]) will be applied to the case database. While other CBR researchers (mainly when dealing with technical diagnosis) heavily rely on this, the assumption of a very weak domain theory again requires a modified strategy. Either the weak domain theory will prevent the detection of rules leading directly to a diagnosis for a given description[8], or these rules will be applicable only in very restricted circumstances, i.e. only for very few problem cases. Furthermore, it might be desirable to extract weaker regularities — heuristics —, too. This is somewhat difficult when using the generalization based methods.

The idea is, not to learn production rules or decision trees, but to derive *determination rules* as described in [9]: How certain attribute values influence other attributes of the case or, possibly, the diagnosis. An example of such a determination rule has already been shown in section 3: If somebody wants to go on a bathing holiday he'll have to go to the sea. On the other hand, a general rule enabling the system to determine the destination for a given type of holiday can not be derived.

However, this part of CABATA is currently being investigated and it is not yet clear *how* these determination rules can be learned efficiently.

## 7 Discussion

### 7.1 Relation to other work

Due to the restricted place here only a few authors can be mentioned — far away from being complete.

The two systems that were of particular importance for the development of CABATA will be discussed in the next sections.

#### 7.1.1 The CcC+-system

The CcC+-system ([15, 14]) employs a quite similar idea of using rules to modify the importance of case attributes. Since the domain is not technical diagnosis, an adaption of the way rules are integrated was necessary. Furthermore, the following modifications lead to a much more powerful architecture:

1. Not only the diagnosis of the reference case is taken into account — rather all attributes of a case may trigger rules.
2. The restricted similarity schemes provided in CcC+ have been designed much more powerful in CABATA by using context graphs (see section 4).
3. While the problem of automatic acquisition of classification knowledge in CcC+ remains completely unsolved, CABATA at least in some situations explicitely asks for this kind of knowledge (section 6.1).

#### 7.1.2 The PROTOS-system

The PROTOS-system ([13]), was also motivated by a weak domain theory disabling a pure model-based approach. To compare different feature values, PROTOS, too, uses matching knowledge to be given to the system in a predefined language:

1. Featural importances are used similar to the way rules may be used within CABATA (see section 3).
2. Structural knowledge explains relations among different attributes of a case. This is similar to the use of rules as constraints (see section 3).

However, both systems differ significantly in the way this knowledge is applied. While CABATA allows interactions between different case attributes (*determinatiuon rules* — see section 6.2), PROTOS focusses mainly on the target classification, for example (taken from [13]):

[7] This is due to the assumption of an existing domain theory in EBL.

[8] In fact, if this were not the case, the whole problem could be solved using the rule-based approach only.

ad 1: A **seat** is essential for the class **chair** while **wheels** are spurious.
ad 2: A **pedestal** enables the **chair** to hold a person, as do **legs**.

In the latter case PROTOS additionally employs a much richer language describing functional relations. This might be integrated in CABATA by enriching the context graph (section 4) with semantic relations. However, these will be highly domain-dependent.

The attempt to learn from a classification by letting the user explain the discrepancies is extended in CABATA by checking whether previously defined constraints were violated. What's more, in PROTOS the explanations are only used to justify a classification while in CABATA domain knowledge may be tought this way.

Another difference between both systems is the structure of the classification: While CABATA constructs a proposal, PROTOS simply chooses one of previously defined diagnostic categories. This, too, is the main reason for the above mentioned differences: The task of CABATA is *problem solving*[9] while PROTOS is used to *classify* problem cases.

### 7.1.3 Other research

Concerning the problem of learning of matching knowledge AHA et. al. ([1, 2]) suggest various algorithms to be applied in a case-based reasoning context. However, for all algorithms a *concept description* is required. Thus these algorithms are hardly applicable for weak-theory domains.

BARLETTA and MARK [8] suggest *Explanation Based Indexing* as a method to use domain-specific knowledge for an efficient memory management. This could be of use in future work on CABATA. However, it has to be extended to be more flexible and context-sensitive.

Another project dealing with technical diagnosis is the MOLTKE workbench ([3, 4, 5, 6, 7, 17]). Here, too, context graphs are used to represent experience knowledge and an attempt is made to combine various sources of knowledge and inference strategies. However, it is not clear, whether all results can be simply adapted to other domains (e.g. decomposition of tasks, use of causal knowledge).

RISSLAND and SKALAK ([18]) employ a *mixed paradigm approach*: Their CABARET system consists of two co-reasoners, a rule-based and a case-based one. Both are capable of running in a stand-alone manner — in CABARET they work together using an agenda-based controller. Though CABARET is applied to legal reasoning, it should be well-suited for the technical domain, too: There *are* rules, underlying causalities etc., and the case-based part could enable the system to work more efficiently by envoking the right part of the rule base. Again, how this approach could be applied to weak-theory domains remains an open question — here the rule-based part is much to restricted to be capable of running as a stand-alone machine.

## 7.2 Future work

Future work on the CABATA-system will include

- improvement of the algorithms used to propagate similarities through context graphs to avoid certain anomalies,
- research about the possibilities to integrate Machine Learning techniques to allow learning beyond the storage of cases, i.e. on the level of domain knowledge,
- integration of methods for efficient memory organization and case retrieval (e.g. indexing).

# References


[1] D. W. Aha. Case-based learning algorithms. In R. Bareiss, editor. *Proceedings 3rd DARPA Workshop on Case-Based Reasoning*, 1991.

[2] D. W. Aha, D. Kibler, and M. K. Albert. Instance-based learning algorithm. *Machine Learning*, 6, 1991.

[3] K.-D. Althoff, A. De la Ossa, F. Maurer, M. Stadler, and S. Weß. Case based reasoning for real world applications. SEKI-Report SR-90-23, University of Kaiserslautern, 1990.


[9] That is, construction of a solution case.

[4] K.-D. Althoff, F. Maurer, and R. Rehbold. Multiple knowledge acquisition strategies in MOLTKE. SEKI-Report SR-90-21, University of Kaiserslautern, 1990.

[5] K.-D. Althoff, F. Maurer, R. Traphöner, and S. Weß. Die Lernkomponente der MOLTKE-Werkbank zur Diagnose technischer Systeme. *KI*, 5(1), 1991.

[6] K.-D. Althoff and S. Weß. Case-based knowledge acquisition, learning and problem solving for diagnostic real world tasks. In *Proceedings EKAW*, 1991.

[7] K.-D. Althoff and S. Weß. Case-based reasoning and expert system development. SEKI-Report SR-91-16, University of Kaiserslautern, 1991.

[8] R. Barletta and W. Mark. Explanation-based indexing of cases. In *Proceedings AAAI*, 1988.

[9] T. R. Davies and S. J. Russel. A logical approach to reasoning by analogy. In *Proceedings IJCAI*, 1987.

[10] O. Etzioni. A structural theory of explanation-based learning. Technical report, Department of Computer Science, Carnegie-Mellon University, 1990.

[11] S. Minton. *Learning Search Control Knowledge - An Explanation-based Approach.* Kluwer Academics, Boston, 1988.

[12] T. M. Mitchell, R. M. Keller, and S. T. Kedar-Cabelli. Explanation-based learning: A unifying view. In J. W. Shavlik and T. G. Dietterich, editors, *Readings in Machine Learning.* Morgan Kaufmann, San Mateo, 1990.

[13] B. W. Porter, R. Bareiss, and R. C. Holte. Concept learning and heuristic classification in weak-theory domains. *Artificial Intelligence*, (45), 1990.

[14] F. Puppe. *Problemlösungsmethoden in Expertensystemen.* Springer Verlag, 1990.

[15] F. Puppe and K. Goos. Improving case based classification with expert knowledge. In *Proceedings GWAI.* Springer Verlag, 1991.

[16] J. Quinlan. Induction of decision trees. In J. W. Shavlik and T. G. Dietterich, editors, *Readings in Machine Learning.* Morgan Kaufmann, San Mateo, 1990.

[17] M. M. Richter and S. Weß. Similarity, uncertainty and case-based reasoning in PATDEX. SEKI-Report SR-91-01, University of Kaiserslautern, 1991.

[18] E. L. Rissland and D. B. Skalak. Combining case-based and rule-based reasoning: A heuristic approach. In *Proceedings IJCAI*, 1989.

# Towards a Case-Based Identification Process *

**E. Paquet, B. Chaib-draa† S. Lizotte**
Département d'informatique, Université Laval
Québec, QC, G1K 7P4, Canada
e-mail: {paquet,chaib}@ift.ulaval.ca

**Abstract**

This extended abstract adresses part of our architecture of individual agent in multiagent environment: the identification process. This process allows to understand the actual situation and select the goal or action that best suits the situation. To achieve this, we are developing a testbed in which the identification is based on past situations of many agents. This type of identification is described in this short paper.

## 1 Introduction

We are currently developing the architecture of an intelligent entity that can evolve in a multiagent environment [CHAI92, CHAI93]. A multiagent environment is a system in which evolve many entities that are more or less intelligent and more or less specialized. These entities are called "agents". To conceptualize the notion of agent, one can think of a factory where several robots realize different tasks individually or in group. These robots can be seen as *agents*. The multiagent theory is derived from distributed AI and may be used in several domains, among which are: distributed problem solving, decision aiding, process control, etc. We presently elaborate the architecture of an agent that would be ideally adaptable to any application domain. To reach this goal, several components must be integrated into the architecture [CHAI93]. Our agent is made up of those components:

- The *Perception* module: it is used to sense the environment.
- The *Identification* module: its purpose is to understand the meaning of the sensed data.
- The *Deliberative* module: it is used to evaluate the consequences of several potential goals or actions, using a cognitive map.
- The *Planning* module: it builds plans to realize the agent's goals.

*This research is supported in part by the Natural Sciences and Engineering Research Council of Canada (NSERC).

†preferred for correspondance.

- The *Action* module: it simply executes the actions commanded by the other components.

Our aim in this paper is to detail the *Identification* module. The task of this module is to understand the situation and select the goal or action that best suits the situation.

We are developing a testbed simulating the behaviour of several vehicles automated with our architecture. In this case, the task of the Identification component is to analyze the perceived situation, which may be composed of several elements like traffic lights, vehicles, walls, etc. According to the situation, the process has to choose among various actions such as slow down, break, turn left, etc.

In a simulated world, it is possible to anticipate every situation that an agent can face. So, in this case, it is feasible to settle the behaviour of an agent with rules like:

```
IF <vehicle in front is too close> THEN <slow down>
or
IF <traffic light is red> THEN <stop>.
```

However, in the real world, the combination of individual elements may produce situations that are very complex and it is quite hard to define an adequate behaviour for every possible situation with such rules.

Therefore, we feel that *case-based reasoning* is more appropriate than rule-based reasoning to fit the needs of the *Identification* module. Case-based reasoning allows to refer to past situations if a specific action is not specifically defined for the current situation. It is not a static process like rule-based reasoning; it allows the agent to acquire experience (that is, learn from new cases) and adapt itself to new situations.

However, case-based reasoning is much harder to implement than rule-based reasoning. Furthermore, the use of case-based reasoning in conjunction with multiagent theory is not a very well documented topic, so we had to take our inspiration from work purely dedicated to case-based reasoning, as [RIES89] [HAMM90] [GOL91] [KOLO93]. Thus, we present in the next sections a design for a case-based reasoner that meets the needs of the Identification module. We named this process *Case-based Identification.*

## 2 The Case-Based Identification Process

As mentioned in the previous section, the Identification module receives as input a situation description and has to select an action that is well suited for this situation.

However, the action is not chosen only accordingly to the situation, but also accordingly to the *mental state* of the agent. The mental state of an agent consists of the knowledge, beliefs and goals of the agent.

For example, if an agent has the goal `reduce fuel consumption` and another agent has the goal `reach destination as fast as possible`, they will react differently if there is a vehicle in front of them: the first agent will probably choose to follow the vehicle in front of him and slow down if needed, while the second agent is very likely going to pass the vehicle. Thus, we see the impact of the mental state on the choice of an action.

We are now ready to describe in detail the case-based identification process, depicted on fig. 1.

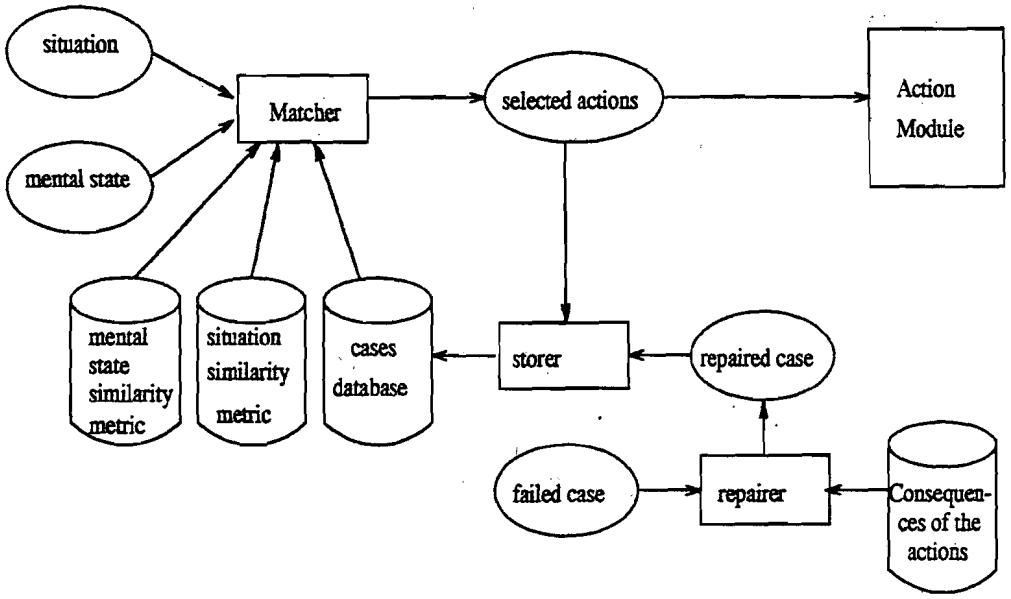



Figure 1: The Case-based Identification Process

The *Matcher* receives as input a situation description and the current mental state of the agent. Its task is to retrieve from the *Cases database* the case that best matches the current input. The cases database initially contains a set of cases obtained experimentally or defined by the designer. A case description is made up of a situation description, a mental state description and the action associated with this case.

Sometimes, the Matcher retrieves a case identical to the current one, but often, there is no identical case in the database. In this situation, the Matcher uses similarity metrics to compare the current mental state and current situation to stored cases. The case that is the most similar to the current case is selected and the action associated with it constitutes the *Selected action.* The new case is then linked to the Selected action and stored in the Cases database by the *Storer.*

The similarity metrics are dependent from the application domain and are determined according to the requirements of the application.

Following the storage of the case, the Selected action is sent to the *Action module* to be executed.

Ideally, the Selected action would always have the expected effects. However, due to the lack of pertinent cases, the Selected action may not be appropriate for the current situation and have undesired consequences for the agent himself or for the community. In those circumstances, the *Repairer* receives the description of a *Failed case.*

The Repairer analyses the Failed case, using a database containing a description of the *Consequences of the actions.* It then tries to select a more suitable action for this case and sends the *Repaired case* to the Storer, which stores it into the Cases database.

But, for the Repairer to receive such a feedback telling him that a certain action went

wrong, someone must have realized that this action had undesirable consequences. This feedback can come from the agent himself or from an other agent. For example, the Selected action may cause effects that go against the own goals of the agent or, for some reason, the agent may be unable to execute the action. In these cases, the agent himself is in position to understand what is going wrong and give the description of a Failed case to the Repairer.

However, since the agent evolves in a multiagent environment, the feedback may come from an other agent. An advantage of evolving with other entities is that you may get experience from more specialized (or more competent) agents. So, if the actions undertaken by an agent harm an other agent or the community in general, the agent in the wrong might receive a message from someone else containing the description of a Failed case.

## 3 Indexing of the Cases

For certain application domains, the Cases database may become huge. So, it is important to use a storage scheme that will allow for a quick retrieval of stored cases and an efficient use of memory. We think that MOPs (Memory Organization Packages) are well suited for this. The concept of MOP has been introduced by Roger Schank as a way to structure and index scenes. He defined a scene as:

> "A memory structure that groups together actions with a shared goal, that occurred at the same time. It provides a sequence of general actions. Specific memories are stored in scenes, indexed with respect to how they differ from the general action in the scene." [SCHA82]

Since this concept of *scene* is very close to our concept of *situation*, it naturally follows that we can use MOPs to adequately index situations. In the same work, we find the following definition for a MOP:

> "A MOP consists of a set of scenes directed towards the achievement of a goal. A MOP always has one major scene whose goal is the essence or purpose of the events organized by the MOPs." [SCHA82]

Thus, we can use MOPs to link situations relating to the same context, that is, situations sharing identical elements. The major scene (or, in our case, major situation) of a MOP contains the elements shared by all the situations related to that MOP. The situations are indexed by their differences from the major situation. This kind of organization provides an almost immediate access to all situations present in memory and an appreciable saving of memory [?].

## Conclusion

In this paper, we presented a method of case-based reasoning that allows an agent to identify a situation in a multiagent environment and act consequently. Using this method, an agent can get experience by storing new cases and can learn from his mistakes through the repair of failed cases.

When we designed the agent architecture, we tried to make it psychologically valid. From this point of view, a process of case-based reasoning fits well in our architecture, since this process stays at a very cognitive level. The concept of MOP, which is used to represent cases in memory, also respects the criterion of psychological validity, since it is based on the functioning of human memory.

We feel that case-based reasoning is a powerful concept and we plan to integrate it into other components of our agent architecture. The planning module, for one, will use a process of case-based planning to elaborate the agent's plans.

## References


[CHAI92] B. Chaib-draa et B. Moulin. Trends in distributed artificial intelligence. *Intelligence Artificial Review*, 6(10), 1992.

[CHAI93] B. Chaib-draa, E. Paquet and L. Lamontagne. Integrating Reaction, Planning and Deliberation in Architecture for Multiagent Environment. *Proc. 3rd Conf. on Computer Generated Forces and Behavior Representation, Orlando, Florida.* 1993.

[GOL91] A. R. Golding and P. S. Rosenbloom. Improving Rule-Based Systems through Case-Based Reasoning. *Proc. AAAI-91.* AAAI, 1991, pp. 22-27.

[HAMM90] K.J. Hammond. Case-Based Planning: A Framework for Planning from Experience. *Cognitive Science,* vol. 14, pp. 385-443, 1990.

[KOLO93] J. Kolodner. *Case-Based Reasoning.* Morgan Kaufmann Pub., 1993.

[RIES89] C.K. Riesbeck and R.C. Schank. *Inside Case-Based Reasoning.* Lawrence Elbaum Associates, Publishers, Hillsdale, New Jersey, 1989.

[SCHA82] R.C. Schank. *Dynamic Memory: A Theory of Reminding and Learning in Computers and People.* Cambridge University Press, 1982.

# Case-Based Reasoning for Network Management

Michael Stadler

OFFIS
Westerstr. 10-12
D-26129 Oldenburg

**Abstract.** Even though today case based reasoning is applied in a wide range of different areas, there are only few systems which make use of case based techniques for network management. In this paper, we outline the domain of network management and highlight consequences for the application of problem solvers operating in this domain. After this, we present a case based prototype performing a task of closed-loop network management upon a simulated computer network together with first results.

## 1 The Domain of Network Management

Network Management covers the operations and strategies for designing, installing, maintaining and operating computer and telecommunication networks. Whereas design and installation of networks both take place off-line, operation and maintenance have to be done during the network's operational phase.

In this brief introduction we will focus upon the latter tasks which are intended to guarantee the desired quality of network services to the user and to collect and evaluate information.

In order to guarantee quality of service, one has to optimize performance, manage configuration and faults and the system has to be kept secure. Information has to be gathered for the purposes of accounting and for gaining information for future network design.

The actions mentioned above all rely on the elementary tasks of monitoring the network's state, reasoning about this state and controlling the network (see fig. 1).

Monitoring provides information about the state of the devices forming the network. After information retrieval, reasoning takes place in order to plan actions to be taken, e.g. for keeping a connection's throughput at a desired level. The reasoning task includes storing information for future use or learning about the network's behaviour. Finally, if the managed network's state has been recognized as optimizeable, undesirable or even critical, adequate control actions have to be taken in order to drive the network back into a desired state. Each of those elementary tasks may be automatted up to a certain degree.

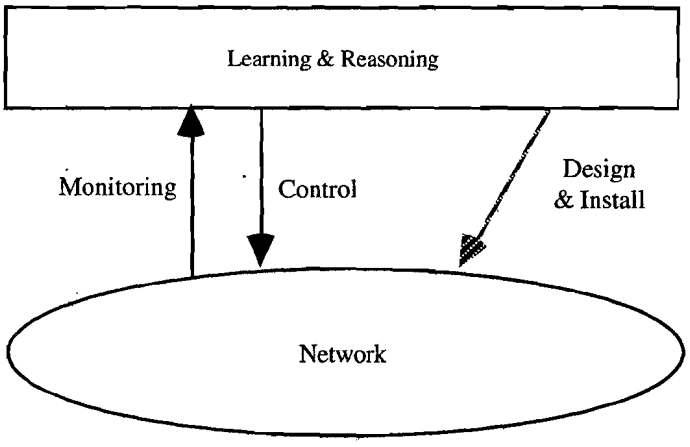



Figure1: Basic actions in network management

In early network installations it was necessary for operators to log in to every single computer system attached to the network for retrieving and changing its network oriented behaviour. Devices without remote access, like hubs, bridges and routers had to be monitored and controlled by lamps and switches or by special control terminals. With this kind of installation, network management could be regarded as an adventure, where both the running and the thinking had to be done by network operators.

Because of growing network complexity the need for systems taking over the "running" part from the network operator emerged. This led to the development of todays management systems. While there are still many problems to solve, those systems provide a uniform access to a large number of different network devices, including computers as well as devices uniquely dedicated to network operation. This is accomplished by making use of standardized management information formats and standardized protocols for the exchange of such information [RFC 1157] [ISO 10040]. Every device that implements an agent, thereby providing access to its management information may be managed by programs taking over a manager role. There exist various structuring principles for management systems consisting of agents and managers, but explaining these would be out of the scope of this paper.

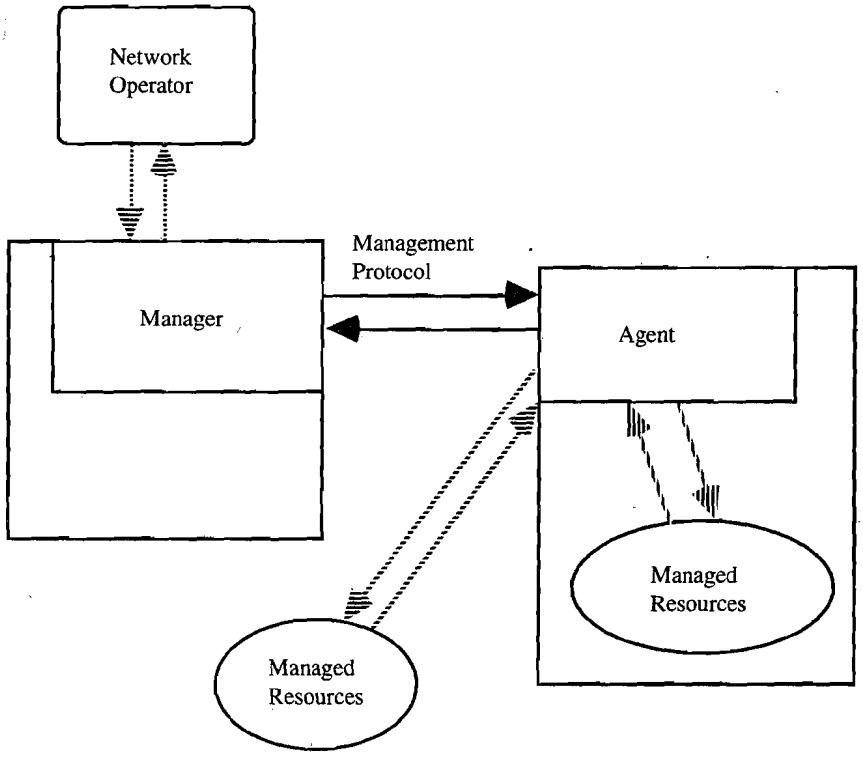



Figure 2: Basic components and structure of a management architecture

Many of today's management systems provide mechanisms giving the operator a better overview of the managed network. They comprise:

- graphical network maps with indication of the site where a problem occurs
- filters which allow for hiding of less interesting network events
- tresholds which can be set for performance parameters which trigger alarms, if exceeded
- execution of shell scripts or simple actions when an alarm occurs in the managed network

Nevertheless, in todays network management systems the reasoning has still to be done by human experts. This task becomes increasingly difficult with growing network complexity and calls for intelligent support.

## 2 Case-Based Reasoning applications in Network Management

Network management is a good domain for application of knowledge based techniques. The knowledge acquisition bottleneck is not as severe as in other domains, given that many network experts have a computer science background and thus may easier express their knowledge in a form adequate for knowledge representation. Even so, it is relatively easy to motivate network operators to test new approaches.

### 2.1 Existing Systems

A number of efforts have already been undertaken to support network operators by means of knowledge based systems [Goyal 91]. Quite a few rule based expert systems for network fault diagnosis, network design and decision support in network topics have been developed so far, but until now there are only two publicated approaches making use of case based reasoning in network management.

NETTRAC [Brandau 91] is a case-based network management assistant. It is concerned with traffic management in telecommunication networks and is designed to advise network operators of problems, and to recommend sets of controls that would alleviate those problems. Cases represent a complete history of a single network problem and the control actions that alleviated the problem.

CRITTER [Lewis 93] is a case-based trouble ticketing system. When an operator solves a network problem, he fills in a so called trouble ticket. The ticket, consisting of a problem description and a solution is entered into a case base and may be retrieved when a similar problem is entered to the system, later on.

The systems mentioned above have in common, that user interaction is mandatory, that is, there are no knowledge based systems which automatically accomplish a closed-loop network management task, performing monitoring, reasoning and controlling.

### 2.2 Requirements for Network Management Expert Systems

The requirements to be fulfilled by an expert system performing closed-loop management are:

- real time response
- work with minimal information
- self control
- easy knowledge acquisition and adaptability
- self adaptation

The task of network management often requires fast reaction on problems for minimizing the effects of network component failures or local bottlenecks. Therefore it is necessary that an efficient reasoning technique comes to use and that it be implemented in an efficient way. Because of good scalability of case based approaches, it seems easier to build a case based expert system that is both fast and compact at a time, than to build a rule based system meeting the same requirements.

Whereas telecommunication networks often have separate lines for passing on management information, in most computer networks the same lines are used for user communication and for passing on management information. Thus, in order to keep the additional overhead of network management small, it is important that an expert system performing closed-loop management solves problems based on as little information as possible. It would be helpful, if the problem solver could deal with imprecise information, thereby allowing to increase the time between information updates.

When operating in a closed-loop mode, i.e. monitoring, reasoning and control tasks are to be accomplished without human interaction during normal operation, it is important that problem solvers operate in a pessimistic manner. That is, control actions have to be verified in case of uncertainty before applying them to the managed network. This can either be achieved by simulations previewing the results of corrective actions or by reporting intended actions to human operators for verification.

A network management system has to operate in a rapidly changing domain. It is thus crucial that the expert system's knowledge base can be easily set up and adapted to new environments. In the domain of network management, it is possible to automatically acquire knowledge by monitoring network operation and by evaluating simulation runs. To make use of these and also to facilitate knowledge acquisition through network experts a simple knowledge representation is needed, likely to be encountered in the area of case based reasoning.

If the environment in which the expert system operates changes, e.g. a new site is connected to a wide area network, changing the network's topology as well as traffic patterns, the expert system must adapt itself to the new situation. This is easily done when using case based techniques relying on graded matches and on treshold values that can automatically be modified.

## 3 A Case-Based Problem Solver for Closed-Loop Network Management

The requirements mentioned in the previous section, together with previous experience in the field of case based reasoning, led us to try a case based approach before investigating other knowledge based techniques for automating network management.

The *ExSim* Prototype which will be described here, consists of three parts. First, there is a simulation program, simulating a wide area network to be managed. The network is composed of gateways which exchange messages, using static routing techniques. Due to this simple routing strategy, local overload may occur decreasing the network's performance, if routing information is not changed by means of management. This task is delegated to a case based reasoner which detects bottlenecks and malfunctions through classification of network states by comparing them to the problem parts of cases stored in its case base.

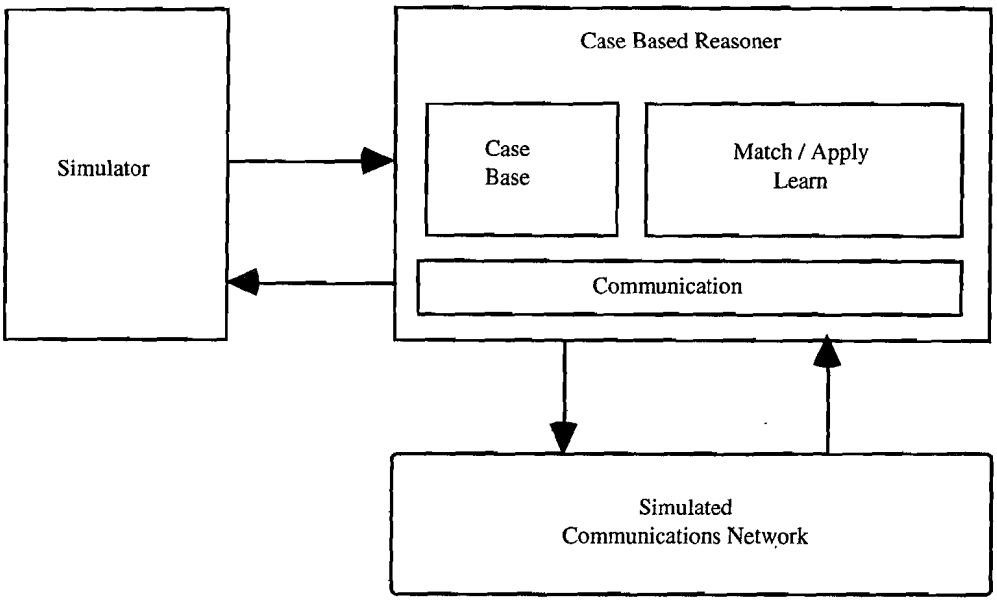



Figure 3: Structure of the ExSim-Prototype, performing closed-loop network management

### 3.1 Knowledge Representation

A case consists of four parts. It is composed of a problem description, a solution description, a unique name and two treshold values, $\alpha$ and $\delta$. Problems and solutions are described by sets of feature/value pairs. Each feature describes an aspect of a possible network component's state.

A problem description consists of a set of gateway routing tables joined into one feature, load information on every network link (i.e. output queue lengths), a topology table and gateway states (i.e. a gateway can be 'up' or 'down'). The domain of the routing table feature is a set of integer matrices, the link load feature's domain is the set of positive floating point numbers and node state features are associated with the domain {'up', 'down'}. Our approach naturally allows for arbitrary domains, but they are not used in the prototypes implementation.

A solution description only consists of a set of routing tables for the managed network's gateways represented by a single feature, like above.

The tresholds $\alpha$ and $\delta$ are used for deciding wheter a case is a candidate for problem solution at all or if a case's solution may be applied to the current network problem, respectively. The property $0 < \alpha < \delta < 1$ is always assured by the system. Whenever a case's similarity to the current problem exceeds its $\alpha$-treshold, it is added to the list of problem solving candidates. If similarity exceeds the $\delta$-treshold, too, following system policy, its solution may be applied to the current problem. Thus, it is possible to influence the probability of cases being chosen for problem solving by adjusting $\alpha$ and $\delta$.

### 3.2 The Similarity Measure

The similarity measure applied for matching cases against network state descriptions is based on the ratio model by Tversky [Tversky 77]. We calculate the ratio between evidences indicating commonalities and all evidences recorded, by means of the function sim, where

$$\text{sim(state, case)} = \frac{a \cdot common}{a \cdot common + b \cdot different} \in [0,1].$$

*common* means the number of features present as well in the network state description as in the case's problem description and whose values are classified as similar. Two values are classified as similar if their similarity exceeds a global treshold t. *different* is the count of features which are present in the network state description and in the case's problem description but whose values are classified as not similar.

Different feature relevances are dealt with by making it more difficult to be classified as similar for values adjoined to highly relevant features than for values adjoined to less relevant features.

At present, network state descriptions always contain the same features as the problem description parts of cases. So it is not necessary to deal with features contained in the cases problem description but not in the network state description, and vice versa. Later on, this will be accomplished with a slight modification to the above function (see [Weß 91]).

To implement a pessimistic strategy, we set the value of coefficient a to 1 and chose 2 as value for coefficient b.

Each feature domain has its own similarity function. Node state values have similarity 1 if either both values are 'up' or both values are 'down'. Else their similarity is 0. Values describing network topology must be identical to be assigned similarity 1, else similarity 0 is assigned. To compute similarity of two routing tables, the number of coinciding entries is counted and divided by the total number of entries in the routing table. Two link load values are similar if they both exceed a treshold C, thereby representing critical link loads, or if they both do not represent critical link loads. C is adjusted according to the maximum and minimum link loads occuring in the network state for guaranteeing specifity of the similarity measure. Thus, link load features are not treated independently by the matcher.

### 3.3 The Problem Solving Strategy

Critical network states are recognized by the reasoner either by receiving a network alarm message indicating an overload in one of the network's gateways and including network state information or by explicitely polling the network state. Network state information consists of a set of gateway routing tables, load information on the network's links (i.e. output queue lengths), topology information and gateway states (i.e. a gateway can be 'up' or 'down'). Upon reception it is compared to the problem parts of cases stored in case memory by means of the similarity measure, described above.

If a matching case is found, the solution contained in the best matching case is sent to the active network components, hopefully alleviating the critical situation. A solution consists of a new set of routing tables for the gateways concerned by the overload or being the source of it.

Should the best matching case's solution already be in use upon occurrence of a network alarm, the case is penalized by increasing its $\alpha$ and $\delta$ tresholds, thereby reducing the case's competetiveness in future matches.

Finding no matching cases may have different meanings, depending on if the problem solver was triggered by a network alarm, or not. In the latter case it indicates, that in terms of the problem solvers knowledge, the network is operating correctly, and no action has to be taken. In the former case, it means, that for an existing network problem there is no solution to be found in case memory. Thus, new knowledge has to be acquired. This is done by passing network state information to a program simulating a network similar to the one being controlled, with the sole difference, that a dynamic load dependent routing strategy (e.g. shortest path routing) is implemented in that network. After the simulation run has ended, resulting in a set of routing tables applicable to the managed network, these are combined with the description of the current network problem, yielding a new case. This case is put into the case memory and its solution part is passed on to the managed network.

### 3.4 First Results

We compared the *ExSim* prototype's performance to the performance of a variant of the shortest path routing algorithm, embedded into the same testing framework. Comparisons were carried out for several different network topologies, as well as for two classes of test scenarios. Class 1 scenarios assumed heavily loaded networks (average load of each link is about 75% of its maximum capacity), class 2 contained scenarios assuming an average network load of 75% maximum capacity with peaks resulting from single batch transmissions.

The results for a network consisting of eight gateways and ten full-duplex links were as follows.

Problem solving with the shortest path routing algorithm was generally about 10 times faster than with the case based reasoner. Implemented in C++ and running on a Sun Sparc 1+ workstation, typical problem solving duration for the case based reasoner was 0.5 seconds if no case had to be learned and 1 second if a new case had to be created. Under the same circumstances typical problem solving duration for the shortest path routing algorithm was 0.08 seconds. Nevertheless, the case based reasoner with a simulation program serving as knowledge source kept the managed network stable, almost whenever the shortest path algorithm did (We call a network stable, if local overloads may be alleviated by rerouting and average link load does not increase over time, given that traffic characteristics do not change substantially). In about 10% of these cases, network behaviour wasn't as good as after solving the same problem by directly using the shortest path algorithm. This is due to the increased problem solving duration as well as to the unverified graded match applied for case retrieval.

To solve by retrieval 60% of the problems occurring in a class 2 scenario with three batch jobs inserted to the network at different times, a case-base with about 100 cases is necessary. A larger number of cases is necessary,

to solve by retrieval the same amount of problems in a class 1 scenario. The reason for this behaviour is, that in class 1 scenarios all kinds of problems are equally like to occur whereas occurring problems are much more specific for class 2 scenarios. Note, that because we didn't implement a mechanism for discarding cases, the case base always tends to grow over time. However, only about one third of cases contained in the case base are frequently reused, so that a significant improvement can be made here.

Whereas the ExSim prototype meets real-time requirements imposed by the particular test domain, performs well for the intended purpose and adapts well to changes, some of the requirements for network management expert systems are not taken into account, at all. The implemented prototype doesn't verify the appropriateness of solutions before applying them to the network. Also, in the approach chosen, complete state information for every network component is needed for problem solving. In a real system this would lead to an enormous overhead by network management traffic dramatically decreasing network capacity.

## 4 Conclusion

There are many application areas for case based techniques in the field of network management. Test results of the prototype described in this paper show, that case based problem solvers may even be efficient enough to perform tasks of closed-loop management of computer networks. This may be especially valuable, when solving problems which can't be solved by standard algorithms.

## 5 References


[Brandau 91] Richard Brandau, Alan Lemmon, Carol Lafond, *Experience with Extended Episodes: Cases with Complex Temporal Structure* in: proc. DARPA Workshop on case-based reasoning, Washington, 1991

[Goyal 91] Shri K. Goyal, *Knowledge technologies for evolving networks* in: proc. IFIP TC6/WG6.6 Second International Symposium on Integrated Network Management, Crystal City, Washington D.C., 1991

[Lewis 93] Lundy Lewis, A case-based reasoning approach to the resolution of faults in communication networks, in: proc. IFIP TC6/WG6.6 Third International Symposium on Integrated Network Management, San Francisco, 1993

[OSI 10040] International Standard ISO/IEC 10040:1992(E). Information technology – Open Systems Interconnection – Systems management overview.

[RFC 1157] Request for Comments 1157, *A Simple Network Management Protocol* , DDN Network Information Center, SRI International, May 1990

[Tversky 77] A. Tversky, *Features of Similarity* in: Psychological Review, Vol. 84, pp. 327-352, 1977

[Weß 91] Stefan Weß, PATDEX/2: *Ein System zum adaptiven, fallfokussierenden Lernen in technischen Diagnosesituationen*, SEKI-Working-Paper SWP91/01, Dept. of computer science, University of Kaiserslautern, Germany, 1991

# Case-Based Reasoning in a Simulation Environment for Biological Neural Networks

Oliver Wendel
University of Kaiserslautern
Dept. of Computer Science
P.O. Box 3049
67653 Kaiserslautern
wendel@informatik.uni-kl.de

**Abstract.** This paper presents a case-based simulation environment devised to assist neurophysiologists in the design and analysis of simulation experiments with biologically oriented neural networks. We describe the problem domain and our specific notion of a case, discuss the complex structure of such cases and present a method to automatically transform the numerical raw data derived from simulations into a symbolic behavioral description that can be used for further inferences.

## 1 Introduction

*MOBIS* - Modeling of Biological Systems - is a case-based, interactive simulation environment devised to assist neurophysiologists in the design and analysis of simulation experiments with biological neural networks. In such a complex problem domain the problem parameters are highly interdependent and solutions are experimental setups fine-tuned through an iterative process of design, simulation, and analysis. Utilizing existing solutions for new problems and for the comparison of simulation experiments thus becomes an interesting issue. The capture and automated use of this type of problem-solving suggests the use of case-based reasoning (CBR) methods. Although this paper presents an AI application in neurophysiology, we omit an in-depth introduction to biological neural networks, the electro-chemical processes in neurons and synapses that are modelled in our simulator and the like. Instead we assume a basic understanding of these processes and, where necessary, provide sufficient detail along the following sections so that the non-biologist can understand the rest of the paper.

The next two sections discuss the problem domain and the simulation life-cycle. We show where the experience and expertise of a neurophysiologist performing simulation experiments can be assisted by CBR-methods. Our notion of a case in this specific context is described in section 4. Section 5 presents a method to automatically transform the numerical raw data derived from simulations into a symbolic behavioral description that can be used for further inferences by the system itself. In section 6 we briefly show the interpretation of neural behavior as a pattern language and finally we summarize and give an outlook on future activities.

## 2 A Model Neuron

The underlying mathematical model of a neuron that we use in our simulations is the classical cable model as proposed by Hodgkin and Huxley ([8]) and others ([9]). A neuron and its components are interpreted as parts of

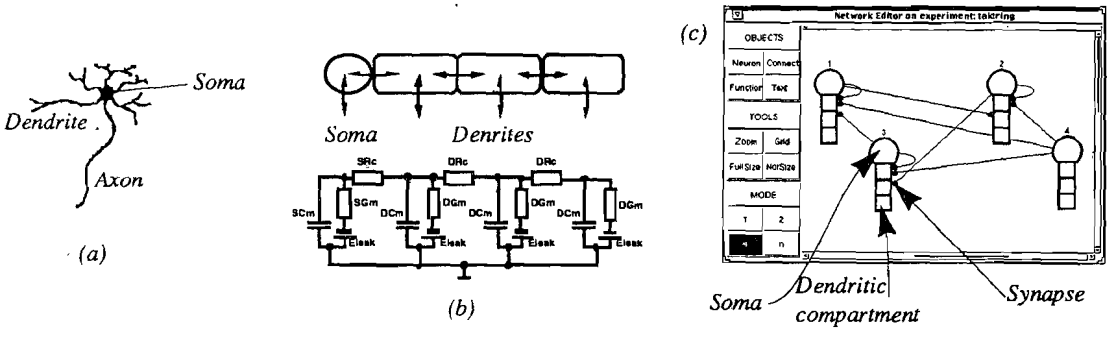



**Fig. 1:** *A neuron (a) and its compartmental model with electrical diagram of the passive properties (b). The corresponding differential equations are numerically computed in each simulation time step. (c) An example network with four neurons created with the interactive graphical network editor.*

an electrical circuit which is described in terms of differential equations that are numerically computed for each simulation time step. Fig. 1 shows an example of such a model neuron and some of its parameters. These

parameters are to be set by the human experimenter in order to achieve a specific behavior of the single neuron. We implemented an interactive graphical editor to design assemblies of such neurons and to compose small neural networks (see Fig. 1).

## 3 Simulation Life-Cycle

In a process called the *design-simulation-analysis-cycle* (c.f. [16]) the human experimenter has to fine-tune a variety of parameters for the network to show a certain, desired behavior.

### 3.1 Design

Typically, he starts off with a baseline experiment, whose outcome reminds him of the desired behavior of the to-be-created experiment. He has to define the topology of the network and the neurons' connectivity, he has to choose values for a multitude of parameters for each neuron and each synapse, and for the whole network. The experimenter also forms hypotheses about the expected result of the simulation, i.e. the expected activity pattern of the neurons. But since many parameter settings are involved in designing a network, exhaustive search on all possible parameter combinations is intractable. At this step, case-based reasoning imitates the use of experience and expertise a human experimenter has acquired: old experiments may have shown interesting outcomes and results that could be exploited in the current situation of designing a new experiment. Thus, experience with old experiments that exhibited similar behavior might be a promising base to start off.

### 3.2 Simulation

Typically, simulations of biological neural networks produce only numerical raw data, as e.g. in the simulation system *GENESIS* ([17]). Our simulation, too, is done numerically by computing the differential equations that describe the network. The behavior of each neuron can be observed by visualizing the numerical outcome as a membrane potential trace.

### 3.3 Analysis

But for a neurophysiologist it turns out that certain *qualitative features* of the simulation (e.g. the presence of *spikes*, or the fact that a neuron remains inactive during a certain period of time whilst another neuron shows activity) represent the main results of a simulation. In this case we would like a computer program to provide (and understand) a representation of the results that includes these qualitative features. Simply graphing the results is helpful but not sufficient for these purposes: a plotting routine does serve to summarize data for the user, but it fails to provide that summarized data in a more abstract and symbolic form that may then be further examined by the computer itself (c.f. [6]). Fig. 2 (a) shows a typical plot of neural activity within a network. We developped

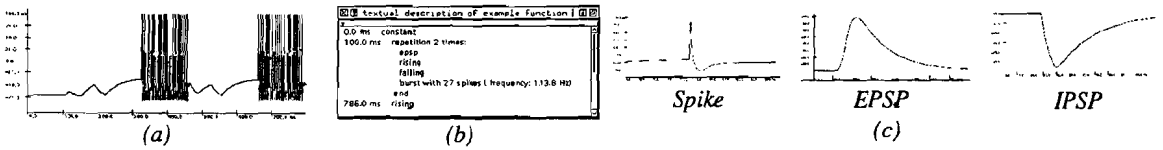



**Fig. 2:** *(a) Neural activity plot with time [msec] and amplitude [mV] axes (the figure shows two subsequent bursts). (b) Automatically generated symbolic textual description of this neural activity pattern. (c) Features in the membrane potential of a neuron. EPSP=excitatory postsynaptic potential, IPSP=inhibitory postsynaptic potential.*

algorithms and a data structure (called *episode structure*) enabling us to represent an overall qualitative description of the results of a simulation or of real digitized experiment recordings. Fig. 2 (b) gives an example of an automatically generated symbolic description which has been textualized. In the analysis phase of the simulation life-cycle, the experimenter has to answer questions like the following:

- Did the network show the desired or expected behavior (*hypothesis evaluation*)?
- Are there any important behavioral patterns within this very special experiment (*intra-experiment analysis*)?
- What are the observable effects of parameter changes along the line of experiment sequences (*inter-experiment analysis, trajectory analysis, sensitivity analysis*)?
- Is it possible to cluster networks or experiments in classes (e.g. oscillators, rhythm generators)?
- Can we identify topological substructures within a complex network that are responsible for certain behavioral aspects?

Aside from having appropriate utilities to graph and statistically interpret the numerical data, the system should supply assistance to answer the questions listed above and propose experiment modifications for a new experiment design. When addressing the problem of automated experiment analysis, the key issue is to construct

a qualitative history of membrane potential plots. This is described later in section 5. The next section discusses the use of CBR in the design and analysis phase and it presents our notion of a case.

## 4 Case-Base Reasoning

*Case-based reasoning* is a general paradigm to reason from *experience* that can be represented as *cases*. It comprises a memory model to represent, index, and organize past experience and a process model to retrieve, integrate, and modify cases. [2] and [10] provide an introductory overview on CBR. [13] gives a comprehensive compilation on actual activities in this area.

### 4.1 General Considerations

Expertise mainly consists of experience. A neurophysiologist doing many computer simulations of neural networks becomes an expert in this domain. He remembers, which experiments he already did and knows about their results. He knows which experiments were successful with regard to a certain aim, which experiments failed, and he has an idea of how to tune parameters in order to validate hypotheses associated with specific networks. Thus, from a CBR point of view, in our domain the notions *case* and *experiment* are identical.

### 4.2 Cases with Complex Structural and Behavioral Component

A *simulation experiment* (or case) consists of a neuronal *structure* and, after running the simulation, the *behavior* of this structure (see Fig. 3 (a) and [5], [14], [15]). The structure comprises the topology of the neural network

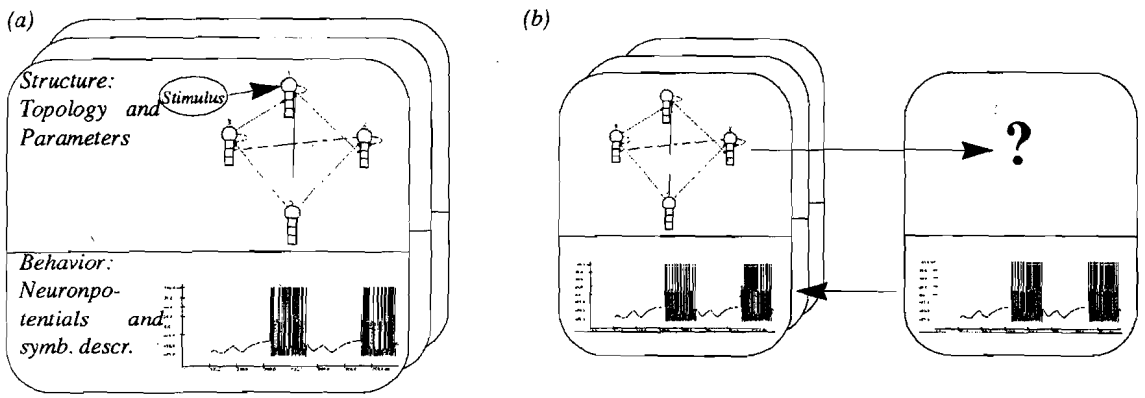



**Fig. 3:** *(a) Simulation experiments as cases. They consist of complex structural and behavioral components. (b) Retrieval of neuronal structures with similar behavior.*

induced by the various neurons and their synaptic connectivity, their specific parameter values (such as capacitances, transmitter release thresholds etc.), and, optionally, stimulus functions applied to a neuron's soma or dendritic compartments. The behavior exhibited by such a network is the computed soma membrane potential traced for each neuron over the whole duration of the simulation. These recordings are transformed into a qualitative description using attributed domain-dependent features which contain further information like duration, amplitude, and frequency. This symbolic description represents the neuronal behavior at a much higher abstraction level than the data-intensive outcome of the numerical simulation and yet is fine-grained enough to capture the most significant features *and* can be further examined and analyzed by the system itself (see sections 5 and 6).

### 4.3 Using Cases

After a simulation run, each experiment (now consisting of the network structure and the qualitative behavioral description) is stored and integrated into a memory structure called *case memory*. Old experiments are used in two distinct ways:

1. *Design*: Prior cases provide a baseline network and set of parameters that are to be modified for new experiments in an iterative cycle of parameter testing, analysis, and parameter adjustment until the desired behavior of a neural network is achieved.
2. *Analysis*: Prior cases are examined to identify network topologies with similar behavior, but possibly very different structure.

A very challenging issue is the analysis of causal relationships between structure and behavior. Digitized membrane recordings of real in-vivo experiments that have been appropriately transformed into a symbolic description could be matched against experiments stored in the case base to identify experimental setups where

neurons exhibited similar behavior. Fig. 3 (b) illustrates this idea. This use of cases will assist the human experimenter by giving hints as to which structure might be found in an organism, given the observable behavior.

## 5 Transformational Steps in Experiment Analysis

The simulation results as obtained by the simulator's output have to be transformed into a symbolic representation. Thus they can be interpreted by the system and used for further inferences (c.f. [6], [11], and [12]). The final representation called *episode-structure* has the following properties: a) It is a qualitative description of a simulation result with descriptional primitives used by the human experimenter. b) It simultaneously realizes data-abstraction and data-compression. c) It can be input to other inference processes. We give a short description of these transformational steps in the subsequent subsections.

### 5.1 Segmentation into Intervals: Scale Space and Interval Tree

For segmenting membrane potential functions into meaningful intervals, various kinds of points (such as extrema of a function and its derivatives) can serve as boundaries. Out of this set of candidates, significant segmentation points are to be selected, generally by application of a digital filter. An automated function segmentation is supposed to extract significant segmentation points and to comply with the following requirements: Omission of noise and unimportant details but preservation of characteristic phenomena, applicability to arbitrarily shaped functions, and significance of interval boundaries based on comparison with the local neighborhood.

These demands introduce the problem of *scale* and impose the use of a variable and adaptive filter parameter, that filters the function at each point with respect to the local neighborhood. Segmenting the function with different scales is achieved by a variable filter parameter and continuuos smoothing. Maxima and minima vanish at a certain scale. Extrema whose scale exceeds a threshold $\sigma$ partition the function into intervals. For different filters $\sigma$, these intervals are subdivided into subintervals so that the whole function can be interpreted as hierarchical tree structure: the root node is the whole function, offsprings represent subintervals with corresponding scales. A stability criterion determines, which segmentation is to be taken.

### 5.2 Feature-Classification

Feature classification is the transformational step where domain-dependent knowledge is introduced for the first time. Features represent typically shaped regions within a function where a domain-specific interpretation can be directly associated with (see Fig. 2 c). Features are detected by a simple rule interpreter, which classifies sequences of function segments according to certain properties (e.g. length, slope, curvature etc.). For different types of functions, separate rule sets are applied.

### 5.3 Grouping into Repetitions

Some phenomena as e.g. *spikes* often appear in packets (this phenomenon is called *burst*). Especially repetitions can be analytically exploited by asking "how does a property of a feature change from one occurrence in a repetition to the next?". Thus it makes sense to think of repetitions of features (or combinations of them) as episodes rather than of single features themselves. Our system finds the shortest possible description in terms of repetitions; these repetitions also can be nested. For example, if $A, B, C$ are features, then the descriptions of the sequences $ABABCABC$ and $AAABCAAABC$ become $AB(ABC)^2$ and $(A^3BC)^2$, respectively.

### 5.4 Symbolic Description

Within our system, the treatment and recognition of repetitions represents the final step towards a "symbolic" function description. As shown in Fig. 2 (b), the description can be visualized in textual form and integrated in an automatically generated analysis report. Experiments frequently are performed in series with slight variation of parameters or conditions. From one simulation to another, potential plots of involved neurons look very similar. Thus the episode sequences can be mapped onto each other, and analogous episodes can be identified. We implemented a matching algorithm for episode structures, which finds a relation with maximum total time of overlapping similar episodes. Two episodes are considered similar, if both are features of the same type or both are repetitions of similar patterns. Thus, e.g. bursts with 5 or 8 spikes can be matched. The differences between identified episodes (e.g. a change in the average frequency of a burst, the strength of a repetition of epsps and so on) are particularly relevant for experiment analysis. Thus the matching algorithm can be used to discover dependencies between experiment parameters and neuron behavior. The user may define formulae constructed of episode parameters. Similar episodes within the experiment series are matched, consequently the variation of the specified formula can be traced automatically, hence supporting the neurophysiologist's analysis. The generated dependency function could be submitted to the same transformation process and be described symbolically. Moreover, it can be used for predicting experiment results by means of correlation analysis.

## 6 Neuronpotentials as Pattern Language?

Another interesting issue we are currently investigating is the interpretation of the symbolic representation of transformed neuron potentials as a sentence $S$ of a pattern language $L$ "spoken" by the neuron. An interesting question thus could be: "What is the underlying grammar $G$ of a neuron's language $L$ with $L = L(G)$" and is it possible to inductively infer this grammar by presenting sufficient example sentences?

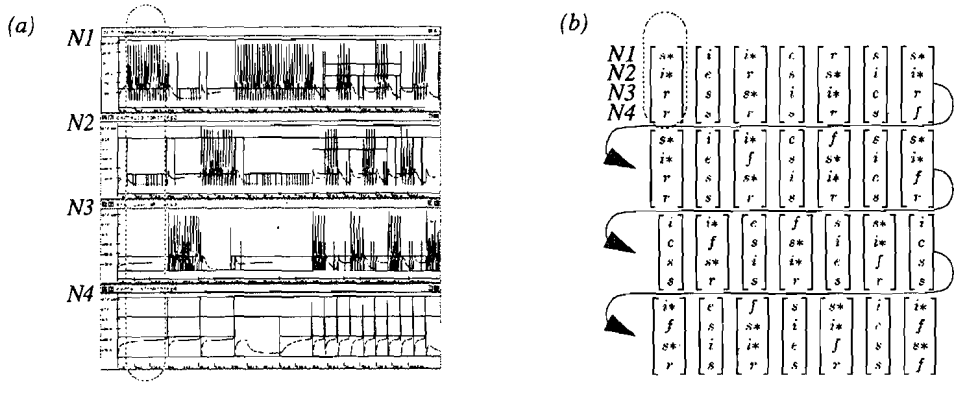



**Fig. 4:** *(a) Simultaneous presentation of the neural activity pattern of all four neurons N1-N4 of the example network. The generated corresponding episode structure is superimposed in this illustration. (b) Pattern description of the neural activity. s=spike, i=ipsp, e=epsp, r=rising, f=falling; *=repetition. Arrows indicate in which direction to read the tuples, the oval around the first tuple in (b) corresponds with the oval in (a).*

The overall behavior of the entire network can be expressed using tokens of vectors consisting of episodes that occur simultaneously in different neurons. In the example of Fig. 4 (a), which shows the neural activity of the 4-neuron network of Fig. 1 (c), the behavior can be described by the pattern in Fig. 4 (b), where each 4-tupel of simultaneous episodes is regarded as attributed character. Several observations can be made using this representation: for example, in the first tuple $[s*, i*, r, r]^T$, a spike in neuron N1 (s*) occurs with an ipsp in neuron N2 (i*). Neurons N1 and N2 exhibit a similar behavior five tuples later, inducing the hypothesis of inhibitory coupling between these two neurons. The characteristic property of this type of network is the existence of three distinct states, where only one of the neurons can fire. Another approach we are currently investigating is the use of tree grammars to describe the resulting episode structure. Fu and others ([4], [7]) propose techniques, how to inductively induce tree grammars from tree examples that are presented to the system.

## 7 Summary and Conclusion

*MOBIS* is a case-based, interactive simulation environment devised to assist neurophysiologists in the design and analysis of simulation experiments with biologically oriented neural networks. In such a complex problem domain the problem parameters are highly interdependent and solutions are experimental setups fine-tuned through an iterative process of design, simulation, and analysis.

A *simulation experiment* (or *case*) consists of a neuronal *structure* and, after running the simulation, the *behavior* of this structure. The structure comprises the topology for the neural network induced by the various neurons and their synaptic connectivity and their specific parameter values. The behavior exhibited by such a network is the computed soma membrane potential traced for each neuron over the whole duration of the simulation. These recordings are transformed into a qualitative description using attributed domain-dependent features. This symbolic description represents the neuronal behavior at a much higher abstraction level than the data-intensive outcome of the numerical simulation and yet is fine-grained enough to capture the most significant features *and* can be further examined and analyzed by the system itself. After a simulation run, each experiment is stored and integrated into the case memory. Prior cases provide a baseline set of parameters that are to be modified for new experiments in an iterative cycle of parameter testing, analysis, and parameter adjustment. The case-based approach is consistent with psychological models of human experimentation performance: expertise and experience are essential in the search for appropriate baseline cases, for the parameter adjustment to meet new requirements, and for the result interpretation. The *MOBIS* system bases its activities on its past experiences and includes the human experimenter in the design-simulate-analyze cycle.

The simulator with the underlying neuron model is fully implemented and can be used as a stand-alone system. It is written in C and runs under OSF/MOTIF on Unix workstations. The simulation evironment is implemented in Objectworks/Smalltalk-80, running on a variety of different platforms. Both systems communicate via files allowing for a shallow coupling of simulator and intelligent experimentation environment.

The system is being developed and used in collaboration with a neurophysiology project in the Kaiserslautern Department of Biology investigating the neurophysiological grounds of the femur-tibia junction and the central flight pattern generator of stick insects (c.f. [3]). Future work will investigate on the applicability of pattern languages and their corresponding grammars such as tree grammars ([4], [7]) in our particular domain. We then would be able to describe, compare and classify neuronal behavior in terms of grammars. A problem still would be the description of temporal relationships between activity patterns in different neurons. Here we will evaluate Allen's time interval relations ([1]). The case memory and appropriate indexing and retrieval structures are currently being defined and will be implemented soon.

## Acknowledgements

We wish to thank all people in the *MOBIS* project. Special thanks are due to Prof. M. M. Richter for his comments and various discussions on our project.

## References

[1] Allen J.: *Towards a general theory of action and time.* In: Artificial Intelligence 23 (2), 123-154, 1984.

[2] Barletta R.: *An introduction to case-based reasoning.* In: AI Expert, August 1991, 43-49, 1991.

[3] Bässler U., Koch U. T.: *Modelling of the active reaction of stick insects by a network of neuromimes.* In: Biol. Cybern. 62, 141-150, Springer Verlag, 1989.

[4] Bhargava B.K., Fu K.S.: *Transformations and inference of tree grammars for syntactic pattern recognition.* In: Proc. Int. Conf. Sys., Man, and Cyb., October 2-4, Dallas, Texas, 1974.

[5] Brandau R., Lemmon A., Lafond C.: *Experience with extended episodes: cases with complex temporal structure.* In: Procs. Case-Based Reasoning Workshop, 1-12, May 1991, Washington D.C., 1991.

[6] Eisenberg M.: *Descriptive simulation: combining symbolic and numerical methods in the analysis of chemical reaction mechanisms.* In: Artificial Intelligence in Engineering, Vol. 5, No. 3, 161-171, 1990.

[7] Fu K.S.: *Sequential Methods in Pattern Recognition and Machine Learning.* Vol. 52 in Mathematics in Science and Engineering, Academic Press, 1968.

[8] Hodgkin A.L., Huxley A.F.: *A quantitative description of membrane current and its application to conduction and excitation in nerve.* In: J. Physiol. (London), 108: 37-77, 1952.

[9] Koch C., Segev I. (Eds.): *Methods in Neuronal Modeling: From Synapses to Networks.* A Bradford Book, The MIT Press, Cambridge, Massachusetts, 1989.

[10] Kolodner J.L.: *An introduction to case-based reasoning.* In: Artificial Intelligence Review 6, 3-34, 1992.

[11] Schrödl S., Wendel O.: *Analysis of neurophysiological experiments using a combination of numerical and symbolic methods.* In: N. Elsner, D. W. Richter (Eds.): Procs. of the 20th Göttingen Neurobiology Conference, S. 736, Thieme Verlag Stuttgart, 1992.

[12] Schrödl S., Wendel O.: *Automated data analysis and discovery in neurophysiological simulation experiments using a combination of numerical and symbolic methods.* In: Procs. Machine Learning ML92 Workshop on Machine Discovery, Aberdeen, Scotland, 1992.

[13] Slade S.: *Case-Based Reasoning: A Research Paradigm.* In: AI Magazine, Spring 1991, 42-55, 1991.

[14] Wendel O.: *Case-Based Reasoning, Experimente und Simulation: Fälle mit komplexer zeitlicher Struktur.* In: Althoff, Wess, Bartsch-Spoerl, Janetzko (Hrsg.): Procs. Workshop Ähnlichkeit von Fällen beim fallbasierten Schließen, 25.-26. Juni 1992, Kaiserslautern, SEKI Working Paper SWP-92-11 (SFB), 163-168, 1992.

[15] Wendel O.: *MOBIS - Ein wissensbasiertes Experimentiersystem zur Simulation biologisch orientierter neuronaler Netze.* In: R. Hofestädt, F. Krückeberg, T. Lengauer (Hrsg.): Informatik in den Biowissenschaften, 1. Fachtagung der GI-FG 4.0.2 "Informatik in den Biowissenschaften", Bonn, 15./16. Februar 1993, Reihe Informatik Aktuell, Springer-Verlag, 203-214, 1993.

[16] Widman L.E., Loparo K.A., Nielsen N.R.: *Artificial Intelligence, Simulation, and Modeling.* Wiley, New York, 1989.

[17] Wilson M.A., Bhalla U.S., Uhley J.D., Bower J.M.: *GENESIS: A system for simulating neural networks.* Technical report, California Institute of Technology, 1990.

# Management Strategy Consultation Using a Case-Based Reasoning Shell

Ansgar Woltering

Department of Computer Science, Technical University of Berlin
D-10587 Berlin, Germany
e-mail: awo@cs.tu-berlin.de

Thomas J. Schult

Department of Psychology, University of Freiburg
D-79085 Freiburg, Germany
e-mail: schult@psychologie.uni-freiburg.de

**Abstract.** Some commercial shells are available to simplify the development of case-based systems. We describe how the process model of case-based reasoning (CBR) is realized by these shells. Furthermore, we examine if a standard real-world and ill-structured diagnostic application can be realized using CBR shells: recommending a strategy type in management consultation. We compare implementations using a CBR shell with a rule-based approach in this domain. Finally, we give a qualitative assessment of four shells with regard to business consultation applications.

## 1 Introduction

Shells can be regarded as an indicator of the degree of generalization achieved in a particular area of knowledge-based systems. To support case-based reasoning, several commercial PC-based shells are available: ART-IM, CBR Express, ReMind, ESTEEM and INDUCE-IT [1,4,5]. After a short look at how the process model of CBR is realized by these shells, we report on our efforts to implement a strategy consultant as a real world test application for the shells.

Common processes in a case-based system can be described by the following cycle [3]:

1. *Input* of a problem description, given a case memory that is not empty
2. *Provision* of several possibly relevant previous cases from the case memory using a similarity measure
3. *Selection* of the most similar case(s)
4. *Adaptation* of these case to the current situation
5. Internal *test* and critique of the adapted solution
6. External *evaluation* and feedback
7. *Learning* by updating the memory or the similarity measure

The first generation shells mentioned above support this process model for CBR in similar ways:

*Input:* Cases are represented as attribute-value pairs, where the values are numerical or symbolical.

*Provision and Selection:* These two steps are replaced by a one step retrieval process in all these shells. Previous cases are selected by nearest neighbor retrieval, counting the weighted results of the comparison of the relevant attributes. For each attribute, predefined criteria for a partial match can be selected. In this way, numbers or texts differing only slightly are recognized as similar. For each attribute one has to specify if it is supposed to influence the similarity of cases. Then, the kind of match is determined, together with the numerical weight of the resulting similarity. Only ESTEEM permits user-defined rules to compare the values and to determine the weight.

*Adaptation:* With the exception of CBR Express, it is possible to adapt a retrieved similar case to the current situation. This may be achieved by using a rule language that allows for an adaptation according to value differences in certain attributes.

The subsequent steps of the process model are not supported by these shells. With regard to functionality in the first four phases, shells can be partitioned in two groups:

- *Simple shells* (ART-IM, CBR Express): flat case structure, feature matching using predefined alternatives, fixed weights.
- *Advanced shells* (ESTEEM, ReMind, Induce-It): hierarchical case structure allowing derived features, user-defined matching procedures, runtime weighting (not all shells), user-defined adaptation.

The goal of our work was to examine if despite these restrictions concerning knowledge representation and process model a standard real-world and ill-structured diagnostic application can be realized using CBR shells.

## 2 The Domain: Management Strategy Consultation

The objective of the domain chosen (management strategy consultation) is a comprehensive evaluation of the ethical values of the management, the capabilities of the employees and the performance of the administrative systems in order to develop a well-suited corporate strategy. In order to fulfill all these requirements the management consultants have to take into consideration a lot of data.

A great deal of work has been done in analyzing general or generic types of company planning situations, and buliding structural models for developing general strategies and focus in any situation [7, 8].
Industry types have been classified for strategic planning purposes as emerging, declining, mature, or fragmented. Within each of these industry types, there are various possible external factors affecting planning, such as the comparative company and competitor position relative to opportunities and critical success factors (which can be affected by buyer and customer strength, by the likelihood of the introduction of substitute products, and by the threat of new entrants), special company markets, organizational and financial factors, and competitors' size and number. Different types of company positions (dominant company, low share of the market, locally concentrated) have also been identified. Based on a study of these characteristics Porter [7, 8] and others have identified a variety of possible generic strategies.

As a consequence, a dependency framework for strategy planning was developed. Dependencies within this framework are often represented as heuristic if-then relationships. For example, if a specific industry type, company, competition, and specific market conditins are given in a situation, then a certain type of general strategy might be worthwhile to consider.

This approach to developing strategies reflects the way many strategic planners work during the initial stages of a project. Additionally, for a preliminary analysis of a situation planners often review their past experience in search for similar patterns that may be useful in solving the situation.

When reviewing a situation, the planner observes, for instance, that the present siuation under study involves a mature industry, where several large competitors are dominant in the market, and where the company being planned for is a relatively small player. The human planner would review any experience with other mature industries to search analogous factors that might suggest possible solution patters, useful to explore in the present. So empirical know-how and experience are the guidance for successful strategic planning, making it a candidate for case-based reasoning, as there doesn't exist a causal model. For a prototype implementation we chose a subtask of the planning process, i.e., the classification of a recommended strategy type of strategic business units (SBU) [7, 8]. Additionally, we try to evaluate the different shells concepts with respect to this classification task.

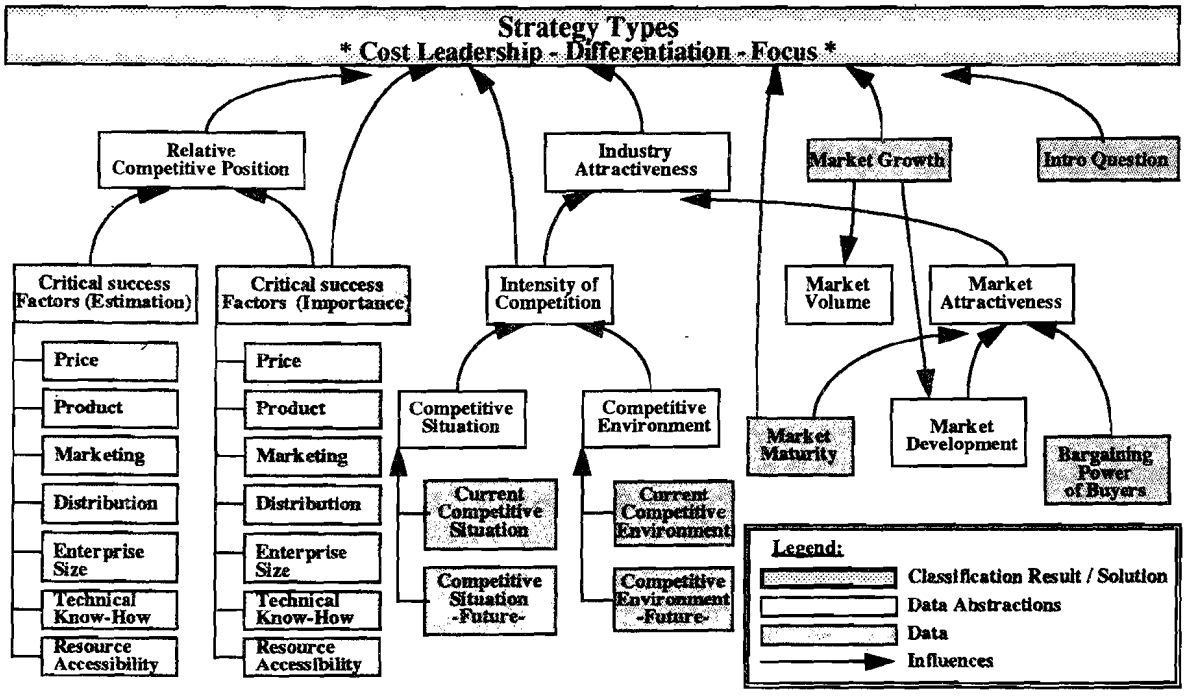



Figure 1. Attribute dependencies in the strategy classification domain

The domain was modeled in a standard rule-based fashion by the system CASA ("Computer Aided Strategy Audit", [2]) that is currently in routine use. CASA analyzes three strategic dimensions that influence the success of an enterprise: corporate culture, market and competitive situation, and strategic cost position. We tried to organize the case-based prototype along CASA's market and competitive situation analysis. Additionally, we built on the CASA system while determining the relevant attributes and their influences on the classification task. Figure 1 shows a model of how to gain strategic recommendations from business data. This general approach provides a useful starting point for developing a prototype case-based decision support system in the strategy planning area. In general, three possible generic strategies can be identified:

- *Cost leadership*: The products or services of the SBU are offered to all or most industrial customers. There is an aggressive investment in productive assets, and a minimization of costs in corporate sectors like Research & Development, service, sales staff, marketing, etc. in order to reach an extensive cost advantage.
- *Differentiation*: The products or services of the SBU are unlike others in the industry. The overall goal is to make them unique in the industry. Customers honour the outstanding benefit to be offered by the products or services.
- *Focus*: In general, the SBU doesn't have a strong position, but it has some special strength in certain products or services. The goal is to focus its resources on the area of its strenght.

The CASA system was implemented using a rule-based expert system shell. Based on the data requested from the user, data abstractions and strategy types shown in Figure 1 are computed by rules.

## 3 Case-Based Versions of the Strategy Consultant: First Results

Our long-term goal is to assess and compare rule-based and case-based approaches in the domain chosen. For a proper comparison, independent judgements on the quality of the respective solutions is necessary. At the moment, we do not have these judgements yet. Therefore, we describe the performance of versions of our case-based alternative named CASTRAC ("Case-Based Strategy Consultation") only with respect to the rule-based system, thus regarding CASA as the reference system.

Several versions of CASTRAC were implemented using the shells ESTEEM and ReMind. By keeping track of the shell features used, we are able to estimate if a given version can be reimplemented using simpler shells. The CASTRAC versions implemented first contain extremely simple similarity measures. They regard all case attributes as equally relevant, without assigning individual weights to the attributes.

CASTRAC's recommendations were compared in three different ways with those of the rule-based CASA system. First order agreement requires the CASA solution to be exactly the strategy type of the most similar case retrieved by CASTRAC. Second order agreement allows the CASA solution to be the strategy type of the most similar or second most similar case retrieved by CASTRAC, third order agreement is defined analogically.

We tested the versions with a case base of 30 consultation cases with solutions generated by CASA. 10 problems had to be solved case-based by CASTRAC. The results are surprising: The simple CASTRAC version performed quite well, resulting in a 80% first order agreement and a 100% second (and third) order agreement with the CASA judgements. As the knowledge engineering complexity of the CASTRAC system was only a small fraction of that of the CASA system, we conclude that for decision support systems simple case-based retrieval mechanisms can be effective if the user is competent enough to assess the cases retrieved.

Furthermore, we tested versions where derived attributes were considered in addition to the problem data. These derived features can be supplied by the user or (as we did) by CASA. However, a certain amount of domain knowledge is needed for an adaquate derivation. However, the quality of the solutions could not be improved by taking into account these additional attributes.

Currently, we are exploring if CASTRAC can be improved by allowing partial matches of attributes. We tried to realize this by ordering attribute values and defining similarity according to this ordering. However, we found the solution quality to decrease, which we cannot explain. Further efforts are needed to accomplish better results.

In addition to that, we are preparing experiments with a human strategy consultation expert in order to assess the quality of the case-based and rule-based version with respect to human experience. As the rule-based version may also be deficient with respect to human judgement, we might have to revise the evaluation results obtained so far.

## 4 Suitability of CBR Shells for Management Support Systems

In the following, we try to give a provisional qualitative assessment of the suitability of four shells with respect to business consultation tasks and with respect to an integration in common software environments. We distinguish between the technical, the organizational, and the functional dimension. The results are shown in Figure 2:

| | ESTEEM | ReMind | CBR-Express | ART-IM |
|---|---|---|---|---|
| **Technical Aspects** | | | | |
| **Flexibility** | 😐 | 😐 | ☹ | ☹ |
| **Speed** | ☹ | 😐 | 😐 | 😐 |
| **Presentation** | ☹ | ☹ | ☹ | ☹ |
| **Tuning** | 😐 | ☺ | ☹ | ☹ |
| **Organizational Aspects** | | | | |
| **Developer** | 😐 | 😐 | ☺ | ☹ |
| **End user** | 😐 | 😐 | ☺ | ☹ |
| **Integration** | 😐 | 😐 | ☹ | ☹ |
| **Functional Aspects** | | | | |
| **Modeling** | 😐 | 😐 | ☹ | 😐 |
| **Cases** | 😐 | 😐 | ☹ | ☹ |
| **Similarity** | ☺ | 😐 | ☹ | ☹ |
| **Adaption** | 😐 | 😐 | ☹ | 😐 |
| **Feedback** | ☹ | 😐 | ☹ | ☹ |
| **Learning** | ☹ | ☹ | ☹ | ☹ |

Figure 2. Qualitative assessments of four CBR shells

Concerning the technical assessment, we take four aspects into consideration:

- *Flexibility*: Is an application easily adaptable to changing demands, e.g., when interface or functionality requirements change?
- *Speed*: How long does it take to transfer, process and present data?
- *Presentation*: Are there tools to present data as common business charts?
- *Tuning*: After first results were obtained, is there a variety of ways to tune an application?

Three organizational aspects are regarded:

- *Developer*: Does creating an application require advanced programming skills? Is it even possible for non-programmers?
- *End user*: Is the user interface difficult to handle
- *Integration*: Can the CBR application be integrated easily in a conventional software environment?

Finally, we assess the shells with regard to six aspects of functionality:

- *Modeling*: Is it possible to model even complex domains?
- *Cases*: What kinds of case structures are possible? Are nested cases and cases of variable length allowed?
- *Similarity*: Is it just possible to choose a similarity measure from a set of predefined building blocks, or can the measure be completely user-defined?
- *Adaptation*: How well does the shell support the adaptation of retrieved cases to the current situation?
- *Feedback*: How difficult is it to integrate user feedback concerning the quality of the solution?
- *Learning*: Beside augmenting the case base, is learning possible, e.g. by an adaptation of the similarity measure with regard to feedback?

INDUCE-IT was left out of the comparison as we did not have enough time to evaluate it due to the difficulty to find a suitable MS-Excel version INDUCE-IT operates with.

We should stress that the gradings for ART-IM refer only to the case-based functionality. In contrary to the other shells, ART-IM is a hybrid development environment for knowledge-based systems with a rich functionality, but offers only basic support for CBR compared to dedicated case-based reasoning shells.

Finally, we tried to aggregate the performance assessments with regard to the three dimensions (technical, organizational, functional). The results are shown in Figure 3.

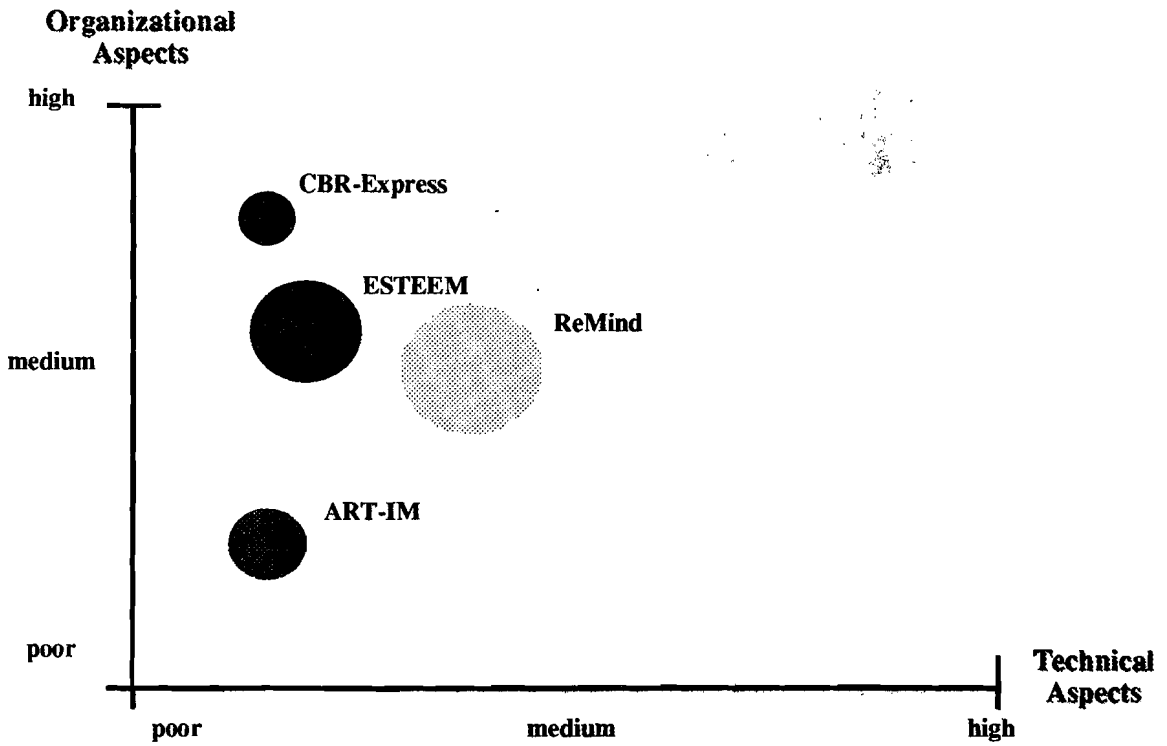



Figure 3. Overview of shell characteristics
(circle diameters correspond to success in functional dimensions)

## 5 Discussion

Schult and Janetzko [6] examine how the case-based process model is realized by these first generation CBR shells and identify elements of the model that are not supported sufficiently, even though generalized methods are known. Based on these shortcomings, they characterize demands on the second generation of case-based expert system shells, in order to make them an adequate environment for complex case-based knowledge engineering.

Here, we just want to add two demands obtained in the management consultation domain:

- The consultant should have the choice whether derived attributes are computed or entered by hand. If they are computed, a flexible language should be provided for that task.
- The system should allow for a change of the similitary measure taking the consultation context into account (e.g., if a quick, but not optimal solution is to be found).

In summary, applications using CBR shells may be a means to ease knowledge acquisition in classification domains. Even without advanced CBR techniques, a prototype with a good performance in the domain of management strategy consultation could be realized. However, CBR solutions might not be as reliable as rule-based or model-based ones. Therefore, we see the role of such a consultant as supporting the expert rather than taking decisions on its own.

## References


[1] Harmon, Paul (1992). Case-Based Reasoning III. Intelligent Software Strategies, VIII, 1.

[2] Krallmann, H., Müller-Wünsch, M. & Woltering, A. (1991). CASA: A knowledge-based tool for management consultants. In: Proceedings of the World Congress on Expert Systems 1991. Pergamon Press.

[3] Rissland, E.L., Kolodner, J.L. & Waltz, D. (1989). Case-based Reasoning from DARPA: Machine Learning Program Plan. In: K. Hammond, Proceedings: Case-Based Reasoning Workshop May 1989. San Mateo, CA: Morgan Kaufmann.

[4] Schult, T.J. (1992). Fallbasierte Expertensystemshells: Methoden und Werkzeuge. University of Freiburg, Dept. of Psychology (Research Reports, No. 89).

[5] Schult, T.J. (1992). Werkzeuge für fallbasierte Systeme. Künstliche Intelligenz 3/92.

[6] Schult, T.J. & Janetzko, D. (i. pr.). Case-Based Expert System Shells: First and Second Generation. In: Proceedings of the Second World Congress on Expert Systems 1994.

[7] Porter, M. (1986). Wettbewerbsvorteile: Spitzenleitungen erreichen und behaupten (Competitive advantage dt.), übers. von A. Jaeger. Frankfurt/Main, New York: Campus.

[8] Porter, M. (1987). Wettbewerbsstrategie: Methoden zur Analyse von Branchen und KOnkurrenten (Competitive Strategy dt.), übers. von V. Brandt und T. C. Schwoerer. Frankfurt/Main, New York: Campus.