$TA3$: THEORY, IMPLEMENTATION, AND APPLICATIONS OF SIMILARITY-BASED RETRIEVAL FOR CASE-BASED REASONING

by

Igor Jurisica

A thesis submitted in conformity with the requirements
for the degree of Doctor of Philosophy
Graduate Department of Computer Science
University of Toronto

Abstract

Similarity plays a central role in theories of human problem solving and thus is important for artificial intelligence research. Although there are different approaches to similarity assessment, the underlying idea is to classify information according to some features, so that we can use it in similar situations. Depending on the application domain, the task at hand, and user preferences, the relevance of individual features may vary, and so will the similarity of the concepts they represent. It is paramount to know what affects feature relevance and how to represent such information explicitly.

The objective of this thesis is to improve case-based reasoning by: (1) achieving better accuracy during classification; (2) retrieving cases that are more relevant to a given problem; and (3) obtaining scalability with respect to case base size, case and query complexity. We achieve this goal by introducing a new theory of similarity-based retrieval that uses variable-context similarity assessment, and by defining an efficient iterative retrieval algorithm that employs ideas of incremental view maintenance algorithms from database management systems. Context is a parameter of similarity that specifies what attributes are involved in similarity assessment between cases, and what set of values may be considered for these attributes. It defines which aspects of a case are important in a particular situation. We also define a set of operations, namely relaxation and restriction, which enable to control the relevance of retrieved cases.

We evaluate competence, scalability and algorithmic complexity of a prototype system on diverse real-world domains. We show how the proposed similarity measure supports flexible computation by trading off the accuracy or precision of the computation process for time and space resources. In addition, the case representation used supports case base organization so that cases similar in a given context can be grouped into clusters. This representation also lends itself to attribute-oriented discovery, a technique that finds relevant attributes and their values. The discovery process improves the representation by grouping together relevant, removing unneeded or adding essential attributes. Performance evaluation shows how the discovery process improves system's competence. Iterative retrieval of cases is efficiently handled by the adoption of incremental view maintenance algorithms from database management systems. Performance evaluation shows that this approach improves efficiency of case retrieval and thus helps to achieve system scalability with respect to case base size, case representation and query complexity.

Acknowledgments

Preparing a doctoral thesis is a lengthy process, and there are many people that influenced my work directly or indirectly. Foremost, I would like to thank my supervisors, Professors John Mylopoulos and Janice Glasgow. Their encouragement, guidance and suggestions are invaluable. They helped me to get started, and they supported my work all along. They contributed to my research and I am deeply appreciative of that.

David Lauzon influenced my work in the early stages, since the work presented herein on variable-context similarity assessment dates back to the times when he was still with the University of Toronto. We exchanged views on many issues pertinent to similarity-based retrieval and I benefited from these discussions.

Dr. Robert F. Casper helped me gain a lot of insight on *in vitro* fertilization and into complex medical problems. He also provided me with real medical data and helped me in assessing useful tasks for performance evaluation of $\mathcal{TA}3$.

The final phase of my work would not be complete without the financial and moral support from the IBM Toronto Lab, Centre for Advanced Studies (CAS). I would like to thank the CSER team, Gabby Silberman, Pat Finnigan, Steve Perelgut, Ivan Kalas, Scott Kerr, Ian McIntosh, Karen Bennet and Chantal Buttery. By working at CAS, I have gained some insight into software engineering problems. The experience I garnered allowed me to apply variable-context similarity assessment to support diverse users during information retrieval.

I would like to express my appreciation to the University of Toronto and the Department of Computer Science for financial assistance. In addition, my educational experience would be limited if I did not have the opportunity to travel to conferences and meetings. For that, I greatly appreciate the travel assistance I received from my supervisors, ITRC, IRIS, PRECARN and Los Alamos National Laboratory.

Last but not least, I would like to thank my parents and my wife. My parents, even though far away, "virtually" supported me all this time. I am especially grateful to my father for introducing me to problems in robotics, namely the inverse kinematics task. My wife Andrea was a source of everlasting support at home. She also contributed immensely to my understanding of the *in vitro* fertilization domain.

Contents

List of Figures v

List of Tables vi

1 Introduction 1

1.1 Motivation . . . 1

1.2 Case-Based Reasoning . . . 1

1.3 Case-Based Reasoning Systems . . . 2

1.4 Performance Issues for Case-Based Reasoning Systems . . . 3

1.5 Problems with Current Case-Based Reasoning Systems . . . 4

1.6 Goals of the Thesis . . . 5

1.7 Thesis Outline . . . 6

2 Literature Review 8

2.1 Case-Based Reasoning Systems . . . 8

2.1.1 Theoretical Foundation of Case-Based Reasoning . . . 10

2.1.2 Evaluating Case-Based Reasoning Systems . . . 18

2.2 Similarity Assessment . . . 20

2.2.1 Philosophical Background on Similarity . . . 20

2.2.2 Psychological Background on Similarity . . . 23

2.2.3 Computational Background on Similarity . . . 28

2.3 Discussion . . . 37

3 Flexible Similarity Assessment 39

3.1 Motivation for Variable-Context Similarity Assessment . . . 41

3.1.1 Tasks to be Addressed by Variable-Context Similarity Assessment . . . 41

3.1.2 The Role of Context . . . 43

3.2 Knowledge Representation . . . 49

3.2.1 Representation of Cases . . . 50

3.2.2 Representation of Context . . . 51
3.3 Context Transformations . . . 55
3.3.1 Transformations for Context Relaxation . . . 58
3.3.2 Transformations for Context Restriction . . . 60
3.4 Context Operations . . . 62
3.5 Similarity Relation . . . 63
3.5.1 Temporal Aspects of the Similarity Relation . . . 63
3.5.2 Retrieve Function . . . 65
3.5.3 Explain Function . . . 66
3.6 Features of Similarity . . . 69
3.6.1 Distributivity . . . 69
3.6.2 Monotonicity . . . 70
3.6.3 Symmetry . . . 70
3.6.4 Transitivity . . . 72
3.6.5 Reflexivity . . . 72
3.7 Discussion . . . 73

4 Using Variable-Context Similarity **75**
4.1 Introduction to In Vitro Fertilization . . . 75
4.2 Similarity-Based Retrieval for IVF . . . 77
4.3 Case Representation in IVF . . . 77
4.4 The Adaptation Process for IVF . . . 79
4.5 Retrieving Cases in IVF . . . 81
4.6 Discussion . . . 89

5 The $\mathcal{TA}3$ System **90**
5.1 General Architecture . . . 90
5.2 Case Representation . . . 92
5.3 Functional Specification . . . 95
5.3.1 Context Manipulation . . . 95
5.3.2 Similarity-Based Retrieval . . . 96
5.3.3 Similarity-Based Explanation . . . 97
5.3.4 Case Adaptation . . . 99
5.3.5 Case Presentation . . . 99
5.4 Incremental Context Modifications . . . 99
5.4.1 Relevant Research . . . 101
5.4.2 Incremental Context Modification in $\mathcal{TA}3$ . . . 102

5.5 Knowledge Management in the $\mathcal{TA}3$ System . . . 104
5.5.1 Knowledge Acquisition . . . 105
5.5.2 Knowledge Evolution . . . 106
5.5.3 Forgetting . . . 108
5.5.4 Learning . . . 108
5.6 Chapter Summary . . . 109

6 Performance Evaluation **111**
6.1 Evaluation Framework . . . 111
6.1.1 Evaluation Measures . . . 111
6.1.2 Issues to be Addressed . . . 114
6.1.3 Hypothesis and Scenarios . . . 117
6.2 Case Studies . . . 118
6.2.1 Prediction and Knowledge Mining in IVF . . . 118
6.2.2 Inverse Kinematics Task . . . 120
6.2.3 Learning Control . . . 124
6.2.4 Iris Flower Classification . . . 126
6.2.5 Character Recognition . . . 129
6.2.6 Retrieval from a Software Repository . . . 131
6.3 Discussion . . . 141

7 Evaluating the Efficiency of $\mathcal{TA}3$ **144**
7.1 Introduction . . . 144
7.2 The Performance Model of $\mathcal{TA}3$ . . . 146
7.3 Efficiency and Scalability Evaluation of $\mathcal{TA}3$ . . . 148
7.4 Discussion . . . 148

8 Conclusions **155**
8.1 Contributions of the Thesis . . . 155
8.2 Future Work . . . 158

A Appendix **188**
A.1 Case Representation Examples . . . 188
A.2 Case Retrieval Examples . . . 188
A.3 Case Adaptation . . . 192
A.4 Case Base Organization . . . 193
A.5 Relation of CBR to Other Areas in AI . . . 196
A.5.1 Learning . . . 196

A.5.2 Analogy 197
A.6 Application Areas of Case-Based Systems 199
A.6.1 Case-Based Design 199
A.6.2 Case-Based Planning 200
A.6.3 Case-Based Diagnosis 201
A.6.4 Case-Based Explanation 201
A.6.5 Case-Based Classification 202
A.6.6 Case-Based Control 204

List of Figures

3.1 *Two examples of how context affects similarity.* 48

3.2 *Defining context on the basis of a problem case.* 54

3.3 *Viewing a case in a satisfiable context.* 55

3.4 *An un-satisfiable context that cannot be used to view a given case.* 56

3.5 *Relevant set of cases.* 64

3.6 *Naive retrieval algorithm. Context is initialized with the attributes and constraints from the input case.* 66

3.7 *Naive iterative retrieval algorithm. Context is initialized with the attributes and constraints from the input case. Special counters are used to prevent repeating context restrictions and relaxations forever.* 66

3.8 *Explain function algorithm. As defined above, the SetOfCases is s subset of the CaseBase.* 67

3.9 *A modified explain function algorithm. SatisfiabilityValue of a given context measures how many cases in a SetOfCases satisfy a given context. Context specialization is aimed at removing the least frequent attribute-values for a given attribute in the context.* | . | *means set cardinality.* 68

4.1 *Case base browser. IVF case is a set of attribute-value pairs, organized into categories.* 78

4.2 *Retrieval function. Retrieved cases for unmodified context.* 80

4.3 *Retrieval function. Retrieved case after the first iteration of context relaxation.* . . . 80

4.4 *Retrieval function. Retrieved cases after second relaxation of context.* 80

4.5 *Context for the first stage of prediction. Query-by-example editor. The query is formed either by browsing the case base and selecting a particular case or by entering a new case description. In both cases,* $TA3_{IVF}$ *returns cases similar to a given example.* . 82

4.6 *Context for the first stage of prediction. In addition to specifying the case description, constraints on attributes can be defined.* 83

4.7 *Context relaxation by reducing the number of attributes required to match and by enlarging the set of allowable values for an attribute.* 84

4.8 *Conceptual hierarchy for the AGE attribute.* 85
4.9 *Context restriction by enlarging the number of attributes required to match and by reducing the set of allowable values for an attribute.* 86

5.1 *TA3 architecture* 91
5.2 *Generalization and specialization along the is-a hierarchy.* 96
5.3 *Incremental case retrieval algorithm. Context is initialized with the attributes and constraints from the input case. Special counters are used to prevent repeating context restrictions and relaxations forever. Context transformations modify attributes of the least important category first.* 104
5.4 An effect of case diversity in a case base. 109

6.1 *Spheric angular robot.* L_1, L_2, L_3 *are the lengths of the links,* $\varphi_1, \varphi_2, \varphi_3$ *are the angles between these links, which may have different ranges.* 121

7.1 *Cost of retrieval for three context modifications. S/I - standard/incremental strategy; G/R - generalization and reduction.* 149
7.2 *Cost of retrieval for three context modifications. I - incremental strategy; G/R - generalization and reduction.* 150
7.3 *Cost of retrieval for standard and incremental reduction as a function of case base size (*$\mid CB \mid$*) and size of the context (*$\mid Context \mid$*). Ten consecutive relaxations are considered.* 151
7.4 *Cost of retrieval for standard and incremental expansion as a function of case base size (*$\mid CB \mid$*) and size of the context (*$\mid Context \mid$*). Ten consecutive restrictions are considered.* 152
7.5 *Cost of retrieval for standard and incremental generalization as a function of case base size (*$\mid CB \mid$*) and size of the context (*$\mid Context \mid$*). Ten consecutive relaxations are considered.* 153
7.6 *Cost of retrieval for standard and incremental specialization as a function of case base size (*$\mid CB \mid$*) and size of the context (*$\mid Context \mid$*). Ten consecutive restrictions are considered.* 154

A.1 *CHEF's recipe: Beef with green beans.* 188
A.2 *CHEF: case representation.* 189
A.3 *Case representation in Cardie system.* 190
A.4 *PROTOS: case representation.* 191
A.5 *Generating a case from a database record in FindMe.* 191
A.6 *ESPANDA: case representation.* 192

List of Tables

4.1 *Accuracy of predicting hormonal therapy. We present suggested (SV) and actua' (AV) values for the treatment, namely day of human chorionic gonadotrophin administration (DAY_HCG) and the number of ampoules of human menopausal gonadotrophin (NO_HMG).* 87

4.2 *Retrieval function. Similar cases in the second stage of prediction.* 88

6.1 *Accuracy of predicting hormonal therapy. Average and absolute errors between suggested and actual values for the treatment, namely day of human chorionic gonadotrophin administration (DAY_HCG) and the number of ampoules of human menopausal gonadotrophin (NO_HMG). Statistical results with 95% confidence level.* 118

6.2 *The context discovery function. Initial and final contexts for the class of pregnant patients. Initial context may remain unchanged if values for a given attribute satisfy the coverage constraint. New constraint may be created if old values are completely off. A constraint may be shifted to increase the coverage, or the constraint may be shrank to make it stronger.* 120

6.3 *Robot characteristics.* 121

6.4 *The inverse kinematics task - An example of a specific context.* 123

6.5 *The inverse kinematics task - absolute and relative errors on robotic data using simple cross-validation; average over 20 random trials (95% confidence intervals).* 123

6.6 *Servo mechanism domain - Context example.* 124

6.7 *Servo-mechanism domain - Absolute and relative errors on servo-data using simple cross-validation; average over 20 random trials.* 125

6.8 *Improvement of $TA3_{Servo\ 2}$, 6 10-WCV over neural nets + instance-based and plain instance-based approaches.* 125

6.9 *Simple cross-validation - results averaged over 40 and 60 random trials of $TA3_{Servo}$.* 126

6.10 *Iris domain for various case base sizes - classification accuracy on predicting the class of iris plant, using simple cross-validation; average over 10 random trials (95% confidence intervals).* 127

6.11 *The character recognition domain – absolute and relative errors on servo-data using simple cross-validation: average over 20 random trials (95% confidence intervals)...* 131

Chapter 1

Introduction

1.1 Motivation

In many areas of decision making, much of the reasoning process is based on experience rather than on general knowledge. For example, legal reasoning and medical diagnosis use experience in the form of past cases more often than generic rules. Medicine used cases in practice and education even before a case-based paradigm was proposed in computer science. Medical students are trained at the theoretical level and they are also presented with some important, prototypical cases. In the medical practice, a doctor's competence increases with experience, that is with the number of solved cases. Often, seeing two patients as similar medical cases and applying knowledge gained from one to the other can save valuable resources (e.g., unnecessary tests) and still provide professional help. Experts remember positive cases for possible reuse of solutions, but negative cases are also useful for avoiding potentially unsuccessful results. Because cases can be used effectively during problem solving they are an efficient and effective way for acquiring and representing experience.

1.2 Case-Based Reasoning

Case-based problem solving or case-based reasoning (CBR) is a paradigm founded on solving new problems by remembering (representing), retrieving and possibly adapting experience, represented as cases (Kolodner, 1993; Watson, 1997). Informally, a case comprises an input (the problem), an output (the solution) and feedback (an evaluation of the solution). However, the representation of problems, solutions and feedback varies from domain to domain. A case can represent a complete solution (e.g., plan, design), which can be retrieved, adapted and reused

Reasoning with cases involves: (1) accepting a new problem description (a case without a solution and feedback) from a user or another part of the system; (2) retrieving relevant cases from a case

base (past problems with similar input); (3) adapting retrieved cases to fit the problem at hand and producing the solution for the input problem; and (4) evaluating the solution.

CBR is founded on the premise that similar problems have similar solutions. Thus, the goal of case-based retrieval is to find the most similar cases for a new problem, expressed as a query. Similarity among cases is computed as a measure of closeness or relevance. Closeness can be measured in terms of numerical or conceptual distance. The more relevant a case is to a problem, the less adaptation is needed and the more precise the solution. Generally, it is better to retrieve fewer high-quality cases than to retrieve additional less relevant cases that result in a poor solution. A major task in CBR is to measure case relevance, used to retrieve relevant cases.

1.3 Case-Based Reasoning Systems

Case-based reasoning systems constitute a special class of knowledge-based systems. Since their goal is to retrieve from a case base the cases most similar to a given problem, similarity-based reasoning techniques are applicable. As in analogical reasoning, the solution to retrieved cases is adapted to fit the problem at hand (Carbonell, 1983; Hennessy and Hinkle, 1992).

Similarity-based reasoning is a machine-learning paradigm where instances are clustered on the basis of their similarity (Jagadish, Mendelzon and Milo, 1995; Law, Forbus and Gentner, 1994; Tversky, 1977; Vosniadou and Ortony, 1989). CBR systems use similarity assessment to find relevant solved cases and use them to solve problems at hand. Thus, competence, efficiency, and scalability of case-based reasoning systems critically depends on the quality of the similarity assessment algorithm.

Instance-based learning algorithms store training examples and use them to classify new problems (Aha, 1992a; Broos and Branting, 1993; Aha, Kibler and Albert, 1991; Quinlan, 1993; Salzberg, 1991; Scott and Sage, 1992). This approach is similar to CBR in the use of actual instances (cases) during reasoning, and the use of the nearest-neighbor algorithms for similarity assessment. However, CBR systems may also modify existing cases to fit the new problem.

Analogy-based reasoning is founded on a process of finding knowledge similar to a current situation and transforming it by adding, deleting, modifying or re-generating specific parts of it (Carbonell, 1981; Barnden and Holyoak, 1994; Veloso and Carbonell, 1993). This process is similar to CBR in finding similarities between cases and adapting them to fit a new problem.

Reinforcement learning (Sutton, 1992; Ram and Santamaría, 1993b; Thrun and Schwartz, 1993; Kaelbling, 1994) credits a local decision, when a a final solution is successful. Although this approach is mainly used in neural networks, it bears a resemblance to CBR systems. First, CBR systems may prefer more frequent cases to less frequent ones. Second, the weight of a feature describing a case may be increased if that feature is used as a discriminating factor during reasoning.

1.4 Performance Issues for Case-Based Reasoning Systems

There are two main issues pertinent to the performance of reasoning systems: competence and scalability. The former refers to the quality of the solution the system can generate. The latter refers to the relationship between the speed of a system generating an answer, and the case base size and the query complexity.

The competence and scalability of a CBR system directly depends on the quality and quantity of its encoded knowledge. On the one hand, the lack of a sufficient number of cases will make the system less competent and will reduce its problem-solving capabilities. On the other hand, a system's efficiency may degrade because of an unmanageably large case base. When designing a CBR system, one has to avoid the two extremes of a fast and incompetent, or competent but intractable system.

CBR systems are learning systems that continually acquire new knowledge. Knowledge acquisition and other types of learning are essential characteristics of intelligent systems. Although learning is automatic for humans, it is difficult to build learning machines. Despite the variations among existing machine-learning paradigms, they all support changes that improve system's performance (in terms of competence, efficiency, or both) over time (Simon, 1983). This requires that the knowledge be represented in structures that can be modified when new knowledge is acquired (McCarthy, 1958; Buchanan and Wilkins, 1993). Thus, the power of a learning system is greatly influenced by the knowledge organization used.

Cases capture problem-solving processes by representing episodes of these processes. This makes CBR a suitable paradigm not only in domains that can be easily formalized but in hard-to-formalize areas as well. Because knowledge acquisition is supported by learning from experience, CBR systems can supplement weak domain models. This requires representation formalisms which preserve relations among cases and also among their parts.

Machine learning and knowledge acquisition are closely related fields. However, the research communities of these two fields are largely separate. Traditionally, machine-learning researchers have emphasized concept learning from a library of pre-classified training examples. In contrast, knowledge acquisition researchers emphasize a broad range of expert tasks with semi-automatic approaches that have a human expert in the acquisition loop. It is generally acknowledged that no significant progress has been made in the area of knowledge acquisition, mainly because the knowledge acquisition problem can be defined only after the issues of knowledge organization, representation and inference are settled (Buchanan and Wilkins, 1993). CBR systems may use knowledge acquisition for initial case base construction and machine learning for subsequent case base organization.

Case representation and the case base structure affects both competence and efficiency of a retrieval algorithm. There are three main requirements for a case retrieval algorithm: quality, flexibility and scalability. The first ensures that only relevant cases are retrieved. If the reasoning engine uses both relevant and irrelevant information to produce a solution, then quality suffers. If the mix

of relevant and irrelevant cases is presented to the user, then the information overload is higher and the user's confidence in the system is decreased. We propose to achieve high relevance of retrieved cases by supporting similarity assessment with explicitly defined context, i.e., variable-context similarity assessment. The context is defined in terms of constraints on attributes and their values. This approach uses flexible criteria to evaluate case relevance. A flexible retrieval algorithm enables criteria changes during case matching and supports imprecise queries. The proposed similarity assessment approach supports iterative browsing and incremental revisions of imprecise queries by consecutively relaxing or restricting constraints on attributes and their values. In addition, matching criteria stated explicitly in context may change depending on domain, task and user. The scalability of a retrieval algorithm is achieved by retrieving cases efficiently, even when the case representation and queries are complex, and case bases are large. The similarity assessment theory proposed in this theses supports efficient implementation of incremental query modifications, which are needed for scalable iterative browsing.

CBR systems are knowledge-based systems, and as such they must address knowledge engineering, as well as psychological and epistemological issues (Buchanan and Wilkins, 1993). A knowledge engineering issue states that control knowledge should be separated from factual knowledge, and outlines how such knowledge should be managed. A psychological issue asserts that the program should have a model that is well understood. An epistemological issue expresses that findings, evidence, justifications, structure and strategy should be distinguished.

The issues stated above also apply to CBR systems. However, because of the uniqueness of the CBR approach to reasoning, some specific problems have been eliminated and others have emerged. Previous work in CBR was geared mainly towards specific domains. This has resulted in numerous prototypes, and insight into the difficult issues and structure of the CBR paradigm. Nonetheless, little work has been done on the development of general principles and domain-independent systems.

1.5 Problems with Current Case-Based Reasoning Systems

The representation and management issues for CBR are challenging problems because many real-world applications require large case bases and efficient processing. Although the representation and management of cases can be handled in a similar manner to traditional knowledge-based systems, there are particular differences that need to be considered.

The majority of existing approaches to the representation and management of large case bases have not been sufficiently elaborated. It is important to define case representations that support task-dependent views of a case base and enable flexible and efficient retrieval of relevant cases.

A flexible retrieval algorithm should recall relevant cases even if the query is imprecisely specified, the problem-solving task changes, or when there is a need to explore a neighborhood of a retrieved

case. All this requires a task-dependent similarity assessment during similarity-based retrieval. In addition, similarity among cases should involve not only numeric distance measures but also symbolic comparisons. Evolution of a case base must be supported by an extensible case representation and task-dependent case base organization.

There is a need to define performance evaluation approaches applicable to CBR. Although there is some preliminary work available (Aha, 1994), it is difficult to compare different CBR systems because: (1) no standard benchmarks for evaluating the performance of CBR systems have been developed; (2) underlying algorithms are not usually described in sufficient detail; (3) systems operate in different domains; and (4) usually, only small case bases are used. In addition, most of the evaluations are subjective and do not include any statistical measures for their significance.

1.6 Goals of the Thesis

The primary objective of this thesis is to design, implement, and evaluate a generic CBR system. Since retrieval is the core part of a CBR system, we first concentrate on the design of an efficient and flexible similarity-based retrieval algorithm. This suggests the kind of knowledge necessary to represent, and the classification mechanism required to support an efficient system. Similarity assessment also contributes to the design of case-adaptation techniques, since it focuses attention on case differences. We will consider design and implementation of machine-learning algorithms to support knowledge mining and generalization. The implemented system will be evaluated in two different domains – medical and industrial – and will be used to predict information, to plan actions, and to discover new knowledge.

The contributions of the thesis include: (1) a new theory of similarity assessment applicable not only to CBR systems but also to information retrieval; (2) a flexible and efficient representation scheme that accommodates different types of cases; (3) a novel approach to classification of cases using a similarity assessment with explicitly-defined context; (4) a flexible retrieval mechanism supporting both inter- and intra-domain retrieval of similar cases; (5) the inclusion of discovery and generalization algorithms into the CBR architecture; (6) the adoption of view maintenance algorithms from database management systems for CBR that support efficient retrieval of cases; and (7) a performance evaluation of the proposed system on real-world domains, assessing its algorithmic complexity, the competence of the reasoning system, and the scalability of the prototype with respect to the case base size, the complexity of case representation and query complexity.

We aim to extend existing case representation and case retrieval techniques by adopting efficient approaches from related areas. This process will utilize experience from building CBR systems for specific domains. The main objective is increased flexibility, competence, and improved efficiency.

A novel approach to similarity-based retrieval, namely similarity assessment using explicitly-

defined context in terms of matching constraints on attributes and attribute values, may improve the flexibility of case-based reasoning. Defining context for similarity explicitly in terms of constraints achieves two main advantages: (1) it permits the use of CBR when solving multiple tasks; and (2) it supports iterative retrieval by successive relaxation or restriction of constraints on attributes and their values.

Scalability of the CBR process can be enhanced by adapting knowledge base management concepts for case representation and case base management. Furthermore, retrieval efficiency can be improved by adapting incremental view maintenance algorithms from database research.

A performance evaluation of the proposed system shows that the proposed similarity assessment approach improves both competence and efficiency by: (1) achieving better accuracy during classification; (2) retrieving cases that are more relevant to a given problem; and (3) obtaining scalable performance in terms of case base size, case size and query complexity. We show how the proposed similarity measure supports flexible computation, in terms of trading off the quality of the results for time and space resources. In addition, the adopted case representation supports efficient case base organization, so that cases can be grouped into clusters, or equivalence classes, that are similar in a given context. Such representation also lends itself to attribute-oriented discovery, a technique that finds relevant attributes and their values, and improves the representation by either removing unneeded attributes/values or adding necessary attributes.

1.7 Thesis Outline

The thesis analyses CBR systems with a focus on representation techniques and retrieval algorithms. Based on this characterization, (1) we present an effective case representation, (2) we introduce a flexible similarity assessment, and (3) we evaluate the prototype system by assessing its competence, algorithmic complexity, and efficiency on several real-world domains.

Chapter 2 presents a literature review of CBR and similarity assessment techniques. Individual approaches are described with accentuation of their strengths and weaknesses. Chapter 3 proposes a theory for similarity assessment where context is defined explicitly using constraints on attributes and their values. We relate the theory to other approaches, prove its features, and identify its representational requirements. Chapter 4 presents an application of the proposed similarity assessment theory to the medical domain. We show how context affects retrieval of relevant cases and how it can be used to support prediction and knowledge discovery. Chapter 5 introduces $\mathcal{TA}3$, a prototype of a CBR system. We describe an efficient and flexible incremental retrieval algorithm that uses the proposed similarity assessment theory. We also discuss case base management issues, such as case representation and case base organization. Chapter 6 discusses methods suitable for performance evaluation of CBR systems. We evaluate competence and scalability of the proposed CBR system

on several real-world domains. Where possible, we compare these results to performance of other systems. Chapter 7 evaluates algorithmic complexity of $\mathcal{TA}3$. We discuss efficiency of the retrieval algorithm, namely its scalability with respect to case base size, context complexity, case representation complexity, and number of iterations during iterative browsing. Chapter 8 summarizes the study, and characterizes open research issues in the presented areas.

Chapter 2

Literature Review

This chapter introduces CBR systems, describes their principles and problem-solving applications. We discuss research prototypes and industrial applications selected from diverse problem domains.

The emphasis of the review is on similarity assessment techniques, which are discussed from philosophical, psychological, and computational points of view. Individual approaches are described with elaboration on their strengths and weaknesses. Representative approaches are compared on the basis of their foundation and their suitability for specific tasks. This partial history of similarity theories motivates the theory of similarity assessment with explicitly defined context, which is presented in Chapter 3.

2.1 Case-Based Reasoning Systems

Case-based reasoning is a process of remembering and reusing experience in the form of cases stored in a case base. The decision-making process finds similarities between a problem at hand and cases in the case base, and reuses stored solutions to solve new problems. Thus, the quality of reasoning depends on the quality and quantity of cases in a case base. With an increased number of unique cases, the problem-solving capabilities of CBR systems improve while their efficiency may decrease.

If the cases are not sufficiently different, i.e., the cases are highly correlated, the system will be limited in the diversity of solutions it can generate (O'Leary, 1993). If the case base is small, the number of possible solutions is limited as well. This can be compensated by using a more elaborate adaptation strategy. However, experience from machine learning (Minton, 1988a), and more recently from the CBR community (Francis and Ram, 1993; Ram and Santamaría, 1993b; Smyth and Keane, 1995), suggests that the system must either restrict the number of learned cases or extra effort must be put towards the development of storage and retrieval mechanisms in order to cope with the utility

problem[1] (Greiner and Jurisica, 1992; Mooney, 1989).

The major processes common to CBR systems include case storage and case retrieval. Case storage should represent every important feature of the experience in a case base. Representation of cases can be flat or hierarchical. Case retrieval recalls relevant cases from the case base. In addition, the reasoner must evaluate the solution in order to make sure that poor solutions are not repeated along with the good ones; this is also referred to as validated retrieval (Simoudis and Miller, 1990).

Individual case-based reasoning systems differ in decision-making methods. We may identify three types of CBR systems: interpretative, problem-solving and decision-support systems. Interpretative systems compare the new situation to recalled experience. The interpretation is used when the problem is not well understood and when there is a need to criticize a solution. Cases are used to provide justification for solutions, evaluation of solutions when no clear-cut methods are available, and interpretation of situations when their boundaries are not well-defined.

Interpretative systems can be further distinguished into justification, interpretation and prediction systems. Justification is the process of making persuasive arguments (Berman and Hafner, 1993; Rissland and Ashley, 1987a). Interpretation is performed by classification of a concept, based on a fuzzy definition of its borders. If, however, no case is found to be similar enough to a new problem, hypothetical situations can be considered (Ashley, 1990; Bareiss, 1989a). Prediction (or projection) is one of the most important parts of the evaluative component of any planning or decision-making system. If everything about the situation is known, projection is just running known inferences forward from a solution. However, in many cases, not all effects can be predicted. Here, cases provide a way of projecting effects based on what has been true in the past (Goodman, 1989). A natural application for interpretative systems is the American legal system: relevant cases form an argument for or against a case. There is no need to modify the past case, since the problem is solved by finding and presenting the relevant cases.

Problem-solving systems derive solutions to new problems by adapting an old solution for the current situation. Such systems are characterized by the following features: (1) they can learn from its mistakes and thus it will not repeat them; (2) they will keep searching for a better solution, even though one is already available; (3) they will consider available results only after they are modified to fit the current needs. It should be noted that a lack of computational resources may prevent fulfilling the second requirement (Jurisica, 1996) and that adaptation in complex domains may not be possible without user intervention (Hennessy and Hinkle, 1992).

A problem-solving CBR system is applied to problems where simple presentation of past cases does not suffice for finding a solution (e.g., planning, diagnosis, design). In such domains a case needs to be modified to fit a new problem. Some parts may be deleted because they are not needed

[1] Utility problem refers to the fact that a machine-learning algorithm may reduce the scalability because of the added computational complexity. The tradeoff between added advantage and higher cost (more complex search) must be taken into consideration.

for the new problem, some parts may be added, since they are missing in the old case and some parts may be changed in a specific way. For example, in a software design system, the adaptation process may look as follows: *put_on(green box)* is changed to *put_on(blue box)*, or $for(i = 0; i < 100; i++)$ is changed to $for(i = 0; i < 3; i++)$. This process may also rely on generalizations, specializations or on analogical problem-solving (different strategies for modification, usually called adaptation strategies, are discussed in more detail by Kolodner (1993)).

Decision-support CBR systems can use multiple past cases to support problem solving (Aha and Breslow, 1997). These systems are a combination of interpretative and problem-solving systems. The main difference lies in the implementation of retrieval and adaptation techniques.

A reasoning cycle for interpretative and a problem-solving CBR systems are alike. It starts with retrieval and results in solution proposal. The main distinction between interpretative and problem-solving CBR systems is that while the former do not use adaptation at all, the latter rely heavily on it. After this intermediate step, both approaches criticize and evaluate the solution and on the basis of the result, either the final solution is proposed, or additional adaptation is performed. If the retrieved solution does not suffice – either because its low relevance, or because additional solutions are required – the system may alter the original query and retrieve additional cases.

Based on different implementation methods, CBR systems can be classified into six categories: (1) Statistically-oriented systems compute conditional probabilities to express the confidence that the problem can be solved using the previous case (Stanfill and Waltz, 1986). (2) Model-based systems explain the cases in terms of a theoretical model. They determine if a past explanation applies to a new situation (Féret and Glasgow, 1993; Koton, 1988b). (3) Planning- and design-oriented systems represent cases as instantiated plans that satisfy goals or avoid goal conflicts. Individual plans are used as solution templates (Bergmann and Wilke, 1995; Hammond, 1989a; Hanks and Weld, 1992; Navinchandra, Sycara and Narasimhan, 1991; Pankakoski et al., 1991; Tanaka, Hattori and Sueda, 1992; Veloso and Stone, 1995). (4) Exemplar-based systems use cases as exemplars of concepts. Examples (prototypes) are represented as simple instances of cases (Bareiss, 1989a; Fertig and Gelertner, 1991). (5) Adversarial or precedent-based systems use cases (precedents) to justify arguments either for or against the problem (Ashley, 1990; Berman and Hafner, 1993; Gonzalez and Laureano-Ortiz, 1992). (6) Context-based systems use similarity assessment, retrieval and adaptation with explicitly represented context information. The case base is structured using context for more efficient access (Jurisica, 1994; Mylopoulos and Motschnig-Pitrik, 1995).

2.1.1 Theoretical Foundation of Case-Based Reasoning

This section characterizes the main parts of CBR systems and points out strengths and weaknesses of available approaches. In addition, Appendix A.5 discusses the relationship of CBR to other paradigms in artificial intelligence, and Appendix A.6 deals with application areas of CBR systems.

Case Representation

A case represents a specific experience in a particular situation. Generally, a case consists of a problem, a solution and feedback. Depending on the domain and the system, cases can have different forms – different representation, size and content. In addition, they can be organized in various ways.

Different CBR systems use different case representations. The research so far has attempted to accommodate existing knowledge representation formalisms (e.g., predicate notation, rules, semantic networks or frames) in order to transform domain knowledge into an appropriate form.

Case representations can be as simple as a database record as in Battle Planner (Goodman, 1989) or a more complex, frame-based representation as in MEDIATOR (Simpson, 1985). However, any given domain may impose special requirements on the representation formalism used.

CHEF is a simple case-based planning system (Hammond, 1989a). It represents cases as a collection of: (1) the general goal (e.g., to prepare a dish); (2) situation (e.g., available/missing ingredients, time); (3) solution (e.g., plan how to prepare the food); and (4) feedback (e.g., result of food preparation).[2]

A flat case representation is used in (Cardie, 1993a; Cardie, 1993b). A case represents the definition of a single open-class word and the context (see Appendix A.1, Figure A.3). Cases are described by 38 attribute-value pairs. Local context is represented by 20 features (syntactic and semantic knowledge), and the global context is described using 13 features (the state of the parser at the unknown word).

PROTOS is a medical case-based system (Bareiss, 1989a). It represents cases as attribute-value pairs (see Appendix A.1, Figure A.4). In its application domain many of the diagnoses manifest themselves in similar ways, and only subtle variations differentiate them. By using feedback from the user, PROTOS learns these subtle differences.

Generally, a case is defined as an episodic experience. It has to carry experience, since the basis of the CBR paradigm is to reuse past cases during solving new problems. Case-based reasoning is a problem-solving paradigm. Thus, the case representation must include a problem description and experience. A problem description must include the context in which the experience was observed. Context may include the goal or task of the episode, the current situation in the problem-solving environment and constraints. The actual experience comprises a solution and a feedback. The solution may contain an action taken in order to solve the problem, reasoning steps, justifications, alternative solutions, and expected outcome. The feedback may include an outcome of problem solving, explanation of the expectation failure, repair strategy, and alternative solutions used.

A case may record experiences that are different from what is expected (Kolodner, 1993). However, in many real-world domains it is advantageous to represent experience in general, both suc-

[2] Additional information is provided in Appendix A.1, Figures A.1 and A.2.

cessful solutions and failures. This way, successful solutions could be reused, while failures could be avoided.

Recall that a case comprises a problem, a solution and feedback, i.e., a positive or negative experience. Thus, a simple record of facts does not constitute a case. Let us consider an information base consisting of records containing videotapes (Hammond, Burke and Schmitt, 1994; Hammond, Burke and Schmitt, 1996). The records are static – no goal/task, action, or outcome is described (see Appendix A.1, Figure A.5). It is a simple database record containing information about videotapes. To change such records into cases, one needs to add an experience, e.g., a customer browsing this database, his/her goal (e.g., finding a horror movie), a situation context (e.g., Friday night), a strategy (e.g., selected preferences and imposed constraints), and an outcome or feedback (e.g., particular record or customer's dissatisfaction).

Consistently with our belief, the ESPANDA system (Garben, Furnsinn and Ruschkowski, 1995) defines cases as problem descriptions and experiences, where an experience comprises a solution with feedback (see Appendix A.1, Figure A.6).

Effectiveness of solution reuse during case-based reasoning depends on case granularity, i.e., the size of knowledge pieces used to represent it. Since not all components of the case are needed all the time, it is more efficient to access only relevant components of the case. There are two options: representing cases as monolithic structures, and representing them as a collection of smaller pieces of knowledge. In some domains, the decision about case granularity is easy. For example, in medical diagnosis each case may comprise information about a single patient, since it is a relatively small chunk of knowledge, independent of the other cases. However, if the system operates in a continuous domain, such as robotics, it might be difficult to decide what should constitute a case, because feedback on a solution might be considerably delayed. Moreover, one session usually consists of many sub-sessions, which are usually interconnected.

Monolithic representation of large chunks of information has both advantages and disadvantages. It helps keeping information together, which preserves context ties. However, it might not be feasible to create an appropriate problem description. In addition, it may be hard to present cases to the user. Reasoning competence of the system may be decreased by keeping both relevant and irrelevant information together. Since the structure of the case would be flat, it might be impossible for the system to generalize the information. Thus, it is clear that even though large cases are useful, they should not be represented as monolithic structures. Some work on non-monolithic case representations has already been done (Ram and Francis, 1996). We believe that object-based representation languages, such as Telos (Mylopoulos et al., 1990), which support partitioning of both the case base and individual cases, solve many problems while it preserves good features. Since in such a language components of the case are "first-class citizens", the difference between representing large cases and storing small, interconnected, cases is diminished.

Case Retrieval

Case retrieval can be characterized as locating relevant experience. Retrieval consists of recalling past cases and selecting the best subset from them. The relevancy of cases is assessed with respect to the current problem. The objective is to retrieve the most relevant cases and to retrieve only a useful number of them. There are three possible selection methods used to retrieve past experience:

1. Cases are classified according to a pre-selected set of features (Birnbaum and Collins, 1989; Hammond, 1989a; Hunter, 1989; Kolodner, 1983; Owens, 1989). The problem is that one cannot predict in advance every important feature for all future problems.

2. The problem description is filtered prior to retrieval. Thus, only relevant features are used. The problem with this approach is that the salient features have to be somehow selected for each problem: in (Hammond, 1989a) they are predefined, and in (Lauzon and Rose, 1994) they are dynamically specified by the user. Other approaches use information from cases in memory to guide elaboration of the problem case (Leake, 1995; Owens, 1994).

3. Cases are filtered after retrieval (Koton, 1988b; Kolodner, 1989; Rissland and Ashley, 1986; Simoudis and Miller, 1990). All these methods are domain-dependent.

In terms of the difference of representation, retrieval algorithms can be classified as associative, hierarchical, or hybrid. **Associative** retrieval classifies any or all features independently of all the other features (e.g., nearest-neighbor classification (Cardie, 1993a; Gonzalez and Laureano-Ortiz, 1992; Hennessy and Hinkle, 1992; Pearce et al., 1992)). During retrieval, the presence or absence of features (either all or a selected subset of them) is counted – in some cases a weighted sum is computed – and the result is used to judge the overall relevance of the case to the current problem. **Hierarchical** retrieval uses features that are organized into a general-to-specific concept structure (e.g., explanation-based learning or inductive learning used for hierarchy construction). **Hybrid** retrieval is characterized by use of both previous paradigms (e.g., discrimination net (Bradtke and Lehnert, 1988; Cardie, 1993a; Hammond, 1989a; Kolodner, 1983; Rissland et al., 1993)). A somewhat different approach is used in the FRANK system (Rissland et al., 1993); if the initial query is unsuccessful, the CBR-task mechanism alters the particular initial values and resubmits the query. This is similar to the task-oriented retrieval (Lauzon and Rose, 1994). However, Lauzon and Rose's system is more flexible than the FRANK system since it uses a conceptual modeling language Telos (Mylopoulos et al., 1990).

From the description of the above mentioned retrieval algorithms it follows that associative retrieval is desirable if flexibility is required. The hierarchical approach is advantageous if the retrieval task is well-defined. The mixed approach shares the advantages of individual approaches and thus can be applied in both cases.

The most problematic parts of the retrieval process are assessing similarity and retrieving similar cases. Psychological experience tells us that cases can be retrieved strictly on the basis of their surface features (superficial, perceivable, intrinsic), abstract features (derived, deep), or both. In many studies (e.g., (Birnbaum and Collins, 1988; Collins and Birnbaum, 1990; Gentner, 1983; Gentner and Forbus, 1991; Keane, 1988; Kolodner, 1992; McDougal, Hammond and Seifert, 1991; Owens, 1993; Simoudis and Miller, 1990; Thagard et al., 1990; Tversky, 1977; Vosniadou and Ortony, 1989)) researchers studied the existence of all three types of remindings; however, there is disagreement on what features are the most useful and important ones for retrieval (these features are referred to as salient and are context- and task-dependent). Despite different opinions on this issue, the goal is to retrieve the most useful cases and an appropriate number of them. The usefulness of cases depends on the task at hand while the number of required cases depends on the reasoning strategy.

Consequently, it is difficult to decide which of the infinite number of surface features to represent and use during retrieval, and what kind of abstract features should be generated. There is a tradeoff between efficiency and quality of retrieval: the system should: (1) support scalable implementation; and (2) represent a sufficient number of features to achieve good competence. The number of retrieved cases can be controlled by preference heuristics (Kolodner, 1989). **Goal-directed preference** favors cases that address the current reasoning goal. Relevant cases are ordered so that cases with more constraints are used first. **Salient-feature preference** favors matching on features which are more important in a particular situation compared to others. **Specificity preference** favors more specifically-matched cases over less specifically-matched ones. **Frequency preference** favors more frequently-accessed cases over less frequently-accessed ones. **Recency preference** favors more recently-accessed over less recently-accessed cases. **Ease-of-adaptation preference** prefers cases that match on harder-to-fix features over those that are easy to adapt.

In approximating a solution, relevant portions of the selected case are extracted to form a ballpark solution. For interpretative case-based systems, proposing a ballpark solution involves partitioning the retrieved cases according to the interpretations or solutions they predict. For problem-solving CBR systems, proposing a ballpark solution involves presenting the most similar past experience, either with respect to symptoms (surface features) or to goals (abstract features).

A set of general guidelines for proposing a ballpark solution can be constructed on the basis of experience from previous research (Ashley, 1990; Bareiss, 1989a; Gonzalez and Laureano-Ortiz, 1992; Hennessy and Hinkle, 1992; Leng, Buchanan and Nicholas, 1993).[3] The internal structure of the old case, especially the dependencies between different parts of the case, determine focus for the reasoner. Simple adaptations should be considered before more sophisticated ones. The system should minimize the need for external input during adaptation, when existing cases (domain model) could be used instead.

[3] Additional details are presented in Appendix A.2.

The definition of similarity in cognitive science can be found in (Gentner, 1983; Gick and Holyoak, 1980; Holyoak, 1985; Tversky, 1977; Vosniadou and Ortony, 1989). However, it is believed that similarity has not been well defined yet. There is some agreement between the CBR community and cognitive psychology researchers that similarity judgments can be either highly goal-dependent, or highly automatic (based on surface similarity). The importance of features used during retrieval varies with the situation at hand (see Appendix A.4 for more details). For fast, easy-to-use, intra-domain remindings, surface features will be the most useful (quick responses to the environment). For more complex remindings (inter-domain, analogic, or thematic remindings) abstract features may be required, even without the use of any surface features. It was observed that the frequency of relational remindings increases with expertise in a domain; however, even experts use surface similarity in some situations. A case-based reasoning system should use both types of remindings (Gentner and Forbus, 1991; Lauzon and Rose, 1994; McDougal, Hammond and Seifert, 1991; Thagard et al., 1990). Pragmatically, use of certain remindings depends on what features are available, what features are adaptable (Leake, Kinley and Wilson, 1997; Smyth and Keane, 1996).

Case-based reasoning systems use specific past experience in problem solving. When a new problem is posed, the system tries to find the most similar past problem, i.e., a case, and modify its solution to fit the new problem. Similarity in CBR systems affects all aspects of case-based reasoning, namely: similarities of salient features of a new case to features predictive of past cases (this suggests relevant cases for retrieval); similarities of remaining case features to features predicted by a potentially relevant case (this confirms the case's relevance); dissimilarities involving features relevant to the solution of the known case guide the adaptation of the solution to the new situation; dissimilarities which lead to problem solving failures (or non-optimal solutions) trigger learning processes which result in case retention, representation refinement, and the acquisition of additional domain knowledge; varying similarity of two analogs affects complexity of analogical mapping (Bareiss and King, 1989; Keane, Ledgeway and Duff, 1991). More details on similarity assessment approaches and a literature review are presented in Section 2.2.

Case Adaptation

In general, adaptation is the process of fixing up an old solution to meet the demands of a new situation. There are only a few examples where the old solution can be reused without any change: either when the case base is sufficiently large, or when cases are represented with different granularities (fine-grain cases comprise a coarse-grain case; thus, the solution can be composed using fine-grain cases). More often, one must modify either a solution or strategy to get a solution for the current problem. The process of modifying the old example to fit a new problem is called adaptation. At the beginning, the system has to decide what should be adapted in the ballpark solution. Later, the adaptation rules are applied to take the differences between the retrieved case and the input case into

account. The main tasks in the adaptation process are to decide what needs to be adapted, estimate if it is worthwhile to continue adaptation, specify how to diminish the inconsistencies between old solutions and the new needs, and perform adaptation.

In general, case adaptation can be achieved by: inserting something new into an old solution (put-in adaptation), deleting something from the solution (cut-off adaptation), and making a substitution in the retrieved solution. From this viewpoint, we can identify several basic adaptation methods. **Null adaptation** is the direct application of the retrieved solution to the new situation. The psychological evidence for this adaptation is in tasks where the solution itself is simple, even though the reasoning to a solution may be complex.

Parameterized solutions is the best understood structural adaptation method. It uses the comparison of the retrieved and input problem descriptions according to specified parameters. Parameterized adaptation is used in HYPO (Ashley, 1990), PERSUADER (Sycara, 1987), and PLEXUS (Alterman, 1986).

Abstraction and re-specialization is a general structural adaptation technique that abstracts the piece of the retrieved solution, and re-specializes it later. Thus, it results in analogical problem solving. This technique is used in PLEXUS (Alterman, 1986) as a second step in the adaptation process (first, a null adaptation is used, and only if something fails, the system re-plans it using abstraction and re-specialization). A similar approach is used in PERSUADER (Sycara, 1987).

Critic-based adaptation is a structural adaptation that uses critics to debug almost correct solutions (Gonzalez and Laureano-Ortiz, 1992; Hammond, 1989a; Simmons, 1988). A critic checks if a particular combination of features can cause a problem in a plan. If such a feature exists, a specific repair strategy is applied. In CHEF, several domain-specific critic-based adaptation rules are used to modify plans, e.g., deletion of unnecessary steps, insertion of missing steps. It also uses general repair-strategies for separation of interfering steps. Similarly, in PERSUADER (Sycara, 1987) a critic checks if the goals of the various actors are satisfied by the particular solution, e.g., if there is no interference among different goals.

Re-instantiation is a derivational adaptation method. It operates on the plan that was used to generate a particular solution. This may impose a limitation on re-instantiation because re-instantiation of a plan is planning. In MEDIATOR (Simpson, 1985) re-instantiation generates resolutions between two disputants, which allows analogical reasoning.

Most of the systems use domain- and task-specific adaptation rules. For each type of adaptation strategy, one has to designate the knowledge required for its application. However, there are some characteristics common among these strategies.[4]

Individual adaptation strategies can be differentiated using the task differences, knowledge differences and domain dependency. **Task differences** put constraints on the time when the adaptation

[4] See Appendix A.3 for further discussion.

must take place. In the planning domain, adaptation usually occurs during run time since evaluation of the plan is done during execution. In design domains, the evaluation has to take place in the design phase. Thus, adaptation will also occur in this phase. Adaptation of an explanation/diagnosis must be driven by an evaluation of how well the current one fits the existing set of constraints.

Knowledge differences specify the character and amount of domain knowledge required for adaptation. Usually, systems use deep understanding of the dependencies in a domain. An example is a system presented by Goel and Chandrasekaran (1989), where design modification techniques are guided by the internal dependency structure of the plan. The system uses only surface-level techniques to adapt cases. As an example, Kass's system (Kass, 1990) relies on external evaluation to ensure that no impossibilities are introduced to explanation during modification.

Domain independence characterizes the dependence of the system on a particular domain. The system uses only domain-dependent rules for adaptation (Alterman, 1988; Carbonell, 1986; Goel and Chandrasekaran, 1989; Kass, 1990). The initial set of techniques is domain independent; later, some domain-specific rules are incorporated through the explanation of expectation failures (Converse, Hammond and Marks, 1989). Modification techniques are domain independent (Hanks and Weld, 1992; Hinrichs, 1989). The system has a set of meta-goals that span across domains and above them specific transformations are constructed to optimize plans (Collins, 1989).

Taking into account what needs to be adapted, we distinguish two types of adaptation strategies: structural and derivational. **Structural adaptation** applies the adaptation rules directly to the solution stored in a case. As an example, CHEF (Hammond, 1989a) modifies particular recipes, and JULIA (Kolodner, 1987) modifies prior criminal sentences. CLAVIER (Barletta and Hennessy, 1989) uses pieces of other cases to adapt the current case. This approach is more suitable than the rule-based approach, since an extremely large number of rules would be required to represent contextual information (in the case of CLAVIER, this information includes material, shape, and position constraints of individual parts). A similar approach is also used as a basis in structure-mapping analogical problem-solving (Falkenhainer, Forbus and Gentner, 1989; Gentner, 1983).

Derivational adaptation uses rules that produced the original solution to generate the new solution by their re-application (Carbonell, 1983; Veloso and Carbonell, 1993). MEDIATOR (Simpson, 1985) adopts this approach and takes advantage of storing not only a solution with a case, but the planning sequence that contributed to the solution.

Although both adaptation techniques are useful for CBR systems, derivational adaptation has two additional advantages: it needs fewer ad-hoc rules and it can be used across domains.

There are domains where problems are difficult to solve in one chunk, and at the same time difficult to decompose and solve separately because of the strong interaction between pieces. Here, the CBR paradigm provides the possibility to solve the problems at once, and to adapt only those parts of the solution which do not fit the new situation (Hennessy and Hinkle, 1992). Solving the

problem by adapting an old solution enables the problem solver to avoid both dealing with many constraints and breaking the problem into pieces (Kolodner, 1992).

It is beneficial to keep track of near-miss cases and to use hypothetical cases for better adaptation. In the HYPO system (Ashley, 1990; Rissland and Ashley, 1987b), this information helps the reasoner to focus on important features, which, if present, would yield a better solution. In the case of XBE (Pankakoski et al., 1991) hypothetical cases are theories formalized to situation-means-ends forms (the basic structure of a case in XBE). Hypothetical cases are incomplete; thus, they cannot be used in matching to the same extent as actual cases. However, they are used to introduce new design choices to a case-based system and to integrate case- and rule-based reasoning.

In PERSUADER (Sycara, 1987) and CBA (Gonzalez and Laureano-Ortiz, 1992) the adaptation is performed using adaptation critics, which use similarity metrics. This allows for meaningful adaptation, but it is a domain-dependent approach.

Although it is desirable to adapt an old experience to a current problem, sometimes it is either not possible to do so, or recalling cases alone suffices in a particular situation (Jurisica et al., 1998). This may occur in complex and ill-formed domains. Then, the system presents relevant cases as advice and it is up to the user to adapt them as necessary. Because of this fact, it is desirable to have a larger case base than in other systems (Kitano, Shibata and Shimazu, 1993). Such systems are referred to as cooperative case retrieval (Hennessy and Hinkle, 1992; Jurisica et al., 1998) or conversational (Aha and Breslow, 1997) systems, since the task of storage and retrieval is done by machine and the adaptation and feedback is done by a human expert. A system is called a cooperative information system when it supports collaboration among heterogeneous systems (Gaasterland, Godfrey and Minker, 1991; Plaza et al., 1996; Wang, 1996).

2.1.2 Evaluating Case-Based Reasoning Systems

Only recently, has any attention been paid to evaluation of CBR systems (in some cases even the size and the structure of a case base is not reported). Most of the systems are not sufficiently evaluated for two reasons: difficulty of comparing different systems in different domains, and small case bases.

It is hard to compare different systems, since there is no underlying methodology. Moreover, it is not always easy to convert a system from one domain to another. The usual solution is to compare the system to the performance of the human expert (Ashley, 1990; Bareiss, 1989a; Gonzalez and Laureano-Ortiz, 1992; Leng, Buchanan and Nicholas, 1993; Tanaka, Hattori and Sueda, 1992), to the performance of the underlying system (Féret and Glasgow, 1993; Golding and Rosenbloom, 1991; Hanks and Weld, 1992; Koton, 1988b), to the performance of other systems (Bareiss, 1989b; Bradtke and Lehnert, 1988; Bradtke and Lehnert, 1988; Cardie, 1993b; Hanks and Weld, 1992; Krovvidy and Wee, 1993; Skalak, 1993), or to the system's performance before and after learning in novel environments (Ram and Santamaría, 1993b). It should be noted that for many of these systems,

the performance evaluation is only limited, i.e., a small case base is used, only few test cases are presented to the system, no statistical evidence on the obtained performance results is reported, etc.

Existing systems have only a small case base and thus the problem of scaling up is impossible to evaluate. Moreover, having only a few cases does not require or justify the use of elaborate retrieval techniques. In many instances a simple linear traversal would be efficient (unless real-time responses are required). Notable exceptions are the SQUAD system for software quality control (Kitano et al., 1992; Kitano, Shibata and Shimazu, 1993) with 25,000 cases, and the ANAPRON system for the task of pronouncing names (Golding and Rosenbloom, 1991) with 5,000 cases and a case-based system for knowledge acquisition (Cardie, 1993b) with 3,060 cases. It should be noted that in some problem domains, even small case bases suffice. The problem with small case bases is, however, how to evaluate them properly when there is only a small selection of cases.

Because CBR systems are studied as both psychological and computational systems, one should evaluate them accordingly. A psychological model should prove to be useful for explanation of psychological data. A computational system should prove to be useful in the real-world applications. System performance should be evaluated with respect to an underlying system, human experts, other reasoning systems, and different CBR systems. A few researchers proposed general principles for the evaluation of CBR systems (Aha, 1994; Cohen, 1989; Koton, 1988a; O'Leary, 1993). Although many times it is neglected, performance evaluation should prove the system's scalability in addition to system's competence. Since not all systems are efficient in all domains, evaluation should present, and discuss both the circumstances and assumptions under which the method works, or fails to provide the solution. It is required, that the compared systems operate under same conditions, and are solving the same problems. However, since different systems may use diverse evaluation metrics, such as cognitive validity, the amount of time required to solve the problem, or the quality of the solution, a direct performance evaluation may not be possible. Evaluation should also characterize if the system relies on other methods, and discuss the dependency between the CBR system and the underlying method.

System evaluation can include not only performance measure (in terms of competence and efficiency) issues but also verification and validation. Verification is performed to ensure consistency, completeness, and correctness of the system. Even though most of the systems rely on ensured consistency (Kolodner and Riesbeck, 1986; Boström and Idestam-Almquist, 1989; Branting, 1992; Gonzalez and Laureano-Ortiz, 1992; Yang, Robertson and Lee, 1993), there are systems which do not require consistency checking (Fertig and Gelertner, 1991; Kitano, Shibata and Shimazu, 1993). Validation ensures correctness of the final system with respect to user needs and requirements. Even though verification of CBR systems is virtually non-existent, validation of some systems has been reported (Ashley, 1990; Bareiss, 1989a).

In PROTOS (Bareiss, 1989a), accuracy serves as a validation method. It is reported that PRO-

TOS is correct 100% of the time (while COBWEB only 58% of the time and ID3 only 29% of the time). The HYPO system(Ashley, 1990) performed validation by comparatively analyzing four (out of thirty) legal cases. In Battle Planner (Goodman, 1989), 10% of the cases was put aside as a validation set. The system achieved 81.3% accuracy in predicting situations from the validation set.

In the PSSP (Protein Secondary Structure Prediction) system (Leng, Buchanan and Nicholas, 1993) the authors used varying reference protein numbers (1 to 55) and a varying window size (20 to 24) to comparatively evaluate the system's performance. The best results were obtained with 55 reference proteins and a window size of 22. The accuracy on a new data set was 67.3% and predictive accuracy 69.3%. Moreover, if the class type of a protein was known, secondary structure prediction reached 90% accuracy. The system's performance is compared to the actual protein structures. In addition, several similarity matrices and different methods are compared as well.

2.2 Similarity Assessment

Reviewing the literature dealing with similarity assessment, one can notice that there are three main approaches used in studying this problem: philosophical, psychological and artificial intelligence. The first approach focuses on defining the notion of similarity and on defining its place in reasoning theories. The second approach looks at modeling human problem solving in its full complexity, and aims at building psychologically valid systems. In contrast, the artificial intelligence community attempts to build useful systems, even if they function differently from the way humans do.

Taking a computer science approach, one needs to implement similarity assessment algorithms in CBR, pattern recognition, databases, etc. The following statements motivate our study of similarity:

- "The number of attributes shared by plums and lawn-mowers could be infinite." – both weigh less than 1000kg and less then 1001kg, both cannot hear well, both have a smell, etc. (Murphy and Medin, 1985).
- "Any two objects are as similar to each other as any other two objects, insofar as the degree of similarity is measured by the number of shared predicates." (Watanabe, 1985).
- "Any two entities can thus be arbitrarily similar or dissimilar, depending on what is to count as a relevant property." (Edelman, 1993).

It is not an easy or straightforward task to represent objects in a way that supports efficient similarity assessment. Following sections discuss philosophical and psychological explanations of similarity assessment, and computational background on similarity discusses various applications areas and different approaches to implementing similarity assessment algorithms.

2.2.1 Philosophical Background on Similarity

Similarity was first studied by philosophers in the 18^{th} century (Hume, 1947).[5] David Hume saw that if objects are similar in appearance, i.e., **surface similarity**, then they will be attended with similar effects. Thus, from causes that appear similar we expect similar effects.[6] This is also the basis of case-based reasoning systems, provided that one considers not only surface similarity of objects, but other relevant aspects as well.

Hume considers surface similarity to be the only similarity (Hume, 1947). According to him (and as we shall see in subsequent sections, to others) surface similarity is based on readily accessible components of objects. He believed that assessing similarity requires to use only simple sensory attributes of objects. In addition, Hume did not consider different perceptions of these attributes by different subjects or by the same subjects but in different contexts. This approach equates potential (surface) similarity with psychological similarity and thus neglects perceptual capacities of the organisms and assumes common environmental properties. Hume's approach to similarity can be called **the common attribute view of similarity** (Wallach, 1958), since he believed that an object "represents" itself by the common set of features it possesses. On the contrary, as mentioned earlier, not all humans describe objects with the same set of features. This set of features depends on the representation language used and the goal for providing a description. As an example, if one does not have a definition for the concept color, then it would be hard to describe a rainbow. However, describing a rainbow as an optical formation would involve the use of different attributes compared to a situation when a rainbow is described for artistic purposes.

Hume's views on similarity of simple ideas are limited when studied thoroughly. He believed that the degree of similarity of two composite ideas depends on the number of simple ideas they have in common. However, Hume assumed that the similarity between simple ideas is given to us directly.[7]

Hume also observed that all arguments from experience are founded on the similarity we discover among natural objects. According to Hume and Külpe (Hume, 1947; Külpe, 1895), two objects can be treated similarly because they are similar to each other[8] or because both are similar to a third object (we shall return to these issues in Section 3.1.2). In addition, Külpe defines another class of similarity, **partial identity**. Partial identity implies that two objects are only partially recognized as two objects (identities) due to their similarity. For example, if the two stripes of yellow color are very close to each other in shades of yellow then they will be only partially recognized as separate stripes. In contrast to partial identity, **full identity** implies that two objects are perceived as two objects (i.e., they are perceived as two separate identities). Thus, one can distinguish individual objects only if they are different in some aspects. It should also be noted that assuming a richer similarity

[5] In Hume's studies, the term resemblance was used instead of similarity, but this makes no real difference.
[6] There are, however, problems where this principle does not apply.
[7] Wallach critiqued Hume's work (Wallach, 1958); see Section 2.2.2 for more details.
[8] Külpe and Mill define similarity as only a slight difference between objects (Külpe, 1895; Mill, 1829).

than the common attribute view of similarity mentioned earlier, objects may be partially identifiable in one situation or representation, and fully identifiable in another situation or representation. For example, two colors may not be distinguishable for a color-blind person as different shades of grey, yet they may be clearly separable colors.

Smee's work was an early recognition of the intimate interrelation between symbolic logic, similarity by set-theoretic measures, and hierarchical ordering (Smee, 1851). He proposed a machine that could operate a rudimentary propositional calculus to convert assertions of similarity into hierarchical classification schemes (further details can be found in (Gregson, 1975)).

Bain considered similarity an important problem in psychology (Bain, 1855). He introduced the "Principle of Similarity" which states:

> *The tendency to be reminded of past occurrences and thoughts of every kind, through their resemblance to something present, is here termed the Law of Principle of Similarity.*

The "Principle of Similarity" can be viewed as an early case-based approach to reasoning. In order to be reminded of a similar solution, a problem at hand must be similar to past experience. Resemblance is used as an undefined primitive term to define similarity. Similarity is used as one of two principles to explain learning (the other one is contiguity). He proposes to assemble classifications by the notion of similarity. Bain considers identity to be the limiting factor of similarity.

New concepts may be formed upon the operation of similar ideas (Bencke, 1871):

> *Concepts arise when the similarities contained in certain intuitions are thought of together, whether they are excited simultaneously or one after the other in immediate succession.*

Fullerton's ideas (Fullerton, 1890) are similar to Bain's views on identity and similarity. His work opened the way to scale similarity over the range from identity to complete dissimilarity. In his words, "*similarity in the ordinary sense of the word is 'identity in diversity' which implies a recognition of two things as two.*" This can be thought of as a variant of partial identity similarity.

James proposed that two things resemble one another because of their absolute identity in respect to some attribute or attributes, combined with the absolute identity of the rest of their being (James, 1890). His views are useful when one considers similarity of only specific views of objects, as defined in databases (Ceri and Widom, 1991; Acker and Porter, 1992; Bertino, 1992; Attardi and Simi, 1993; Motschnig-Pitrik, 1995), i.e., similarity in a specific context (see Section 3.1.2).

Goodman considers similarity to be a four-place predicate, noting the non-transitivity of similarity matches (Goodman, 1951). An important aspect of his work is that he understood the importance of the question "how similar are the compared objects?"; he states:

> *In keeping with its ordinary use we cannot properly ask whether two qualities are similar*

but only how similar they are or whether they are more or less similar than some other pair.

Using a numerical weight to express similarity of objects partially answers the question posed by Goodman (1951). It is often possible to order objects based on their similarity. However, numerical similarity still does not answer the "how" completely. Using symbolic similarity assessment gives us more insight into how the objects are similar, and it helps to order individual objects on the basis of their similarity.

Carnap refers to a resemblance relation as a similarity relation if it is symmetric and reflexive, and an equivalence relation if it is also transitive (see Section 3.6 for other opinions on symmetricity and transitivity of the similarity relation) (Carnap, 1967). Carnap also introduces **similarity circles**, which are a primitive topological notion of classes of similar things. Tversky suggests an analogous approach, where similar objects can be put into a class (Tversky, 1977).

Summary

The philosophical approaches to similarity described here formed the basis for subsequent psychological and computational studies of this topic. Philosophical ideas can be separated into refutable ideas and ideas supporting our approach to similarity assessment.

Refutable ideas comprise Hume's common attribute view of similarity (Hume, 1947). He shows the limitations of a fixed approach to similarity assessment. Goodman's rendering of similarity as a non-transitive relation (Goodman, 1951) can be refuted, when context is explicitly stated.

Ideas contributing to this thesis include Külpe's partial identity (Külpe, 1895) and Fullerton's identity in diversity (Fullerton, 1890). Their work show that an insufficient representation schema makes objects indistinguishable. Smee's set-theoretic similarity measures and hierarchical orderings of similar objects (Smee, 1851) show the necessity to implement similarity measures. Bain's specification of the principle of similarity (Bain, 1855) supports a CBR paradigm. James's explanation of resemblance of objects by judging their absolute identity with respect to some attributes (James, 1890) supports partial matching to determine similarity. Goodman's understanding of the need to decide how objects are similar instead of merely whether they are or not (Goodman, 1951), supports the similarity assessment with variable context. Carnap's definition of similarity as a symmetric, transitive and reflexive relation (Carnap, 1967) supports results obtained using a pragmatic approach to similarity similarity assessment.

2.2.2 Psychological Background on Similarity

Substantial work on similarity has been carried out by psychologists and cognitive scientists. In this section, we briefly review some of the more relevant literature in this area.

Tversky's work is one of the first approaches to defining a theory of similarity assessment. He defines similarity between objects as the proportion of number of attributes that share a value to the number of all attributes defined for these objects (Tversky, 1977). This forms the **contrast model** approach to similarity assessment. Dissimilarity is defined as a negation of similarity.

Tversky presents several examples where he describes features of similarity (Tversky, 1977). The dependency of similarity on context and frame of reference is also discussed; however, his formalism does not capture this dependency. As a result, not all aspects of similarity assessment can be captured, and as discussed earlier, insufficient representation leads to distorted results. Tversky neither considered nor represented changed or implicitly-assumed context as a phenomenon affecting similarity judgment. His main conclusions are based on a "psychological weight" (or assumed context during the actual comparison) between compared objects. As a result, Tversky concludes that similarity is asymmetric and non-transitive. Tversky's (and other's) psychological experiments on human subjects clearly show asymmetry and non-transitivity. However, the reason for this is not due to the nature of similarity but because of implicitly assumed context. A more detailed discussion of Tversky's approach and enhancement of the representation of similarity is presented in Section 3.1.2.

There is a distinction between independent- and interactive-cue models (Nosofsky, 1990a). The former is based on prototypes and the latter on exemplars. A prototype is the "centroid" of all category exemplars. Independent-cue models assume that category judgment on the basis of overall similarity can be derived from an additive combination of the information from component features. Interactive-cue models, also known as context models, reject the assumption of additivity.

It has been shown that the independent-cue model achieves good performance in predicting categories for exemplars if and only if the categories are linearly separable, regardless of whether multiplicative or additive similarity rules are used (Nosofsky, 1990a). In contrast to this, an interactive-cue model is able to predict categories of exemplars for both linearly and non-linearly separable categories, provided that multiplicative-similarity rules are used.

Next, we present useful measurements of similarities that were shown to be of interest to the research community. There are diverse situations where similarity is used and thus different kinds of similarity might be needed. One of the most common distinctions made is between **surface** and **deep similarity**. The former one is based on easily perceivable attributes of objects; the latter is based on not so readily accessible attributes. It should be noted, however, that the distinction between surface and deep similarity is not clear-cut. What is surface similarity in one context (or in one domain) can easily be deep similarity in another context (or in another domain). Moreover, there is a causal relation between deep and surface similarities (Medin and Ortony, 1989); the former one constrains and sometimes even generates the later one. There is another problematic issue. Namely, objects can be described by different sets of attributes. One of the reasons for this is that not all people perceive objects the same way (Edelman, 1993). For example, a color-blind person perceives

colors as shades of grey, instead of color hues, and an expert can describe a particular object or an event with more detail than a naive user. Another reason is that with a changed situation (i.e., changed context) needs vary as well (Peterson, 1979). For example, describing an apple as an edible object will require a different set of attributes, compared to describing it as an artistic object.

People identify, extract and compile a list of features relevant to a given task using their long-term memory, which Gregson calls **selective emphasis** (Gregson, 1975). In addition, one person's concept of similarity may differ from that of another person (Peterson, 1979).

An **analogical similarity** can be viewed as a form of deep similarity, since more reasoning or more information is required in order to produce an intra- or inter-domain analogical inferences. Wallach defines **psychological similarity** in terms of common responses (Wallach, 1958).

Surface Similarity

Wallach in his effort to revise Hume's work (see Section 2.2.1), defines **potential similarity** as the number of common environmental features that two objects or events display (Wallach, 1958).

Surface similarity is based on readily accessible components of objects (Medin and Ortony, 1989). A similar definition is presented in (Davies and Thomson, 1988, p. 292), where **euphoric similarity** is defined as a similarity of the perceptual or conceptual characteristics of an item and the characteristics of some memory representation. Gentner treats surface similarity as similarity between object attributes as opposed to a similarity between relations (Gentner, 1983; Gentner, 1989). She describes attributes as one-place predicates that characterize simple descriptions of objects, and relations as complex, relational properties of objects.

Rips refers to surface similarity as **perceptual similarity** and introduces it as cognitively primitive and well defined (Rips, 1989). He argues that perceptual similarity can be used as a construct to explain other psychological functions (e.g., categorization). Smith and Heise define perceptual similarity as one that embodies and reflects implicit knowledge (Smith and Heise, 1990). However, in contrast to Hume's fixed approach to similarity, Smith and Heise show that the perceived similarity of two objects changes with selective attention to specific perceptual properties. Edelman defines perceptual similarity as one that is readily perceivable (Edelman, 1993). He proposes that similarity relative to a small but diverse set of prototypes is a natural computationally feasible candidate for a generic representation scheme. He also notes the importance of monotonicity for such a model.

Salient similarity may replace surface similarity, when it uses easily retrievable aspects of representations (Vosniadou, 1989). These aspects do not need to be descriptive attributes. Rather, they can be related to similarity in relational or conceptual attributes, provided that these attributes are easily accessible. Salient similarity enables retrieval of inter-domain analogies. She also suggests that the surface vs. deep similarity distinction is a dynamic one because over time the representations alter and thus may change the salience of attributes. Even though a property can be described as an

object attribute, it does not necessarily mean that it is also represented in people's representation of the particular object or that it is an easily accessible property.

Deep Similarity

Deep similarity is based on more central, core properties of objects (Medin and Ortony, 1989).[9] This is in line with the other experimental results presented (Brewer, 1989). Ross observed that although surface similarity is used more often, structural similarities are more successful in enabling students to solve a new problem (Ross, 1989).

Deep similarity may be viewed as **structural similarity** (Gentner, 1983; Gentner, 1989). Holyoak and Thagard propose that what forms structural similarity are the identities that influence goal attainment (Holyoak and Thagard, 1989). Vosniadou notices that relational properties of objects may be particularly salient even for young children (Vosniadou, 1989). This shows not only that surface similarity can be salient, but that relational properties can also be easily accessible. Moreover, the salience of attributes may change over time.

It was observed that experts can perceive the underlying similarities in problems, whereas novices are not capable of seeing the same similarities mainly because of the difference in surface attributes (Chi, Feltovich and Glaser, 1981). A nice example showing these differences is presented in Vosniadou (1989). Subjects were asked to interpret statements such as: "Clouds are like a sponge." The experimental results show that children mainly used surface features (attributional interpretation) and explained the sentence as "Both are round and fluffy", whereas adults mainly produced relational interpretations and explained the sentence as "Both can hold water and later give it back."

There is a connection between surface and deep similarity. Surface similarity serves two purposes. First, it is a heuristic to find deeper attributes (Medin and Ortony, 1989). Second, it constrains the predicates that compose the mental representation.

Psychological Similarity

Hume thought that there is only surface similarity. In addition, he did not distinguish different perceptions of simple sensory attributes of objects from different subjects from the same subjects but in a different situation. As mentioned earlier, these ideas are limited when studied thoroughly. The reasons for the failure of his definition of similarity are presented by Wallach (1958).

Wallach's work is considered to be the first modern study of similarity, namely **psychological similarity** (Wallach, 1958). Wallach defines psychological similarity in terms of common responses as opposed to Hume's definition in terms of common environmental properties. Wallach pointed out two main reasons Hume's views on similarity are limited. The first one is failure to note the full

[9] This should always be considered in the light of a particular knowledge representation scheme because, as was already stated, objects do not have attributes; the attributes are assigned to them in the representation.

range of attributes in terms of which objects and events can be grouped. The second one is failure to separate potential similarity from psychological similarity.

Wallach studied the psychological aspects of similarity and noted that the presence of a particular attribute is neither a necessary nor sufficient condition for similarity judgments. On the one hand, a particular attribute may be present and yet the similarity judgment may not be made. On the other hand, a particular attribute may be absent and yet the items may be judged similar. This led him to define psychological similarity from four aspects (Wallach, 1958):[10]

(1) *Common environmental properties* (Hume, 1947): In this situation, one can change the similarity by varying the common values of attributes. However, the capacities of the organism to select and ignore attributes are not considered.

(2) *Common responses*: This definition is based on one of the two observations. There are situations where an attribute-value pair is present in two items and yet a different response may exist for them. The attribute-value pair may be absent and yet a common response be made. In other words, if a person responded the same way to two different situations, then those situations are considered psychologically similar. However, according to several experiments, the presence or absence of common responses does not guarantee the presence or absence of psychological similarity.

(3) *Primary stimulation gradients*: This similarity definition is based on measurement of the distance between initial stimulus distance and the old stimulus: the farther they are the less similar items are to each other. One of the best known studies in this area is by Pavlov (1927). .

(4) *Assigning items to a common category*: A particular set of attributes may discriminate items and assigns them to the classes. Varying either the nature of the instances or the classification bias influences psychological similarity in ways one can predict.

Analogical Similarity

Analogical reasoning involves the transfer of relational information from the source to the target. The basic classification of analogies uses the distinction between **intra-domain analogies** and **inter-domain analogies**. It was observed that different similarity assessments should be deployed for these two types of analogical reasoning. For intra-domain analogies, the likelihood of retrieving an analog increases when the surface similarity between the source and the target is increased (Vosniadou and Ortony, 1989). For inter-domain analogies, the retrieval of analogs can be based on salient similarity. Because some relational knowledge may be involved in the retrieval of inter-domain analogies, it was observed that experts and adults have better access to intra-domain analogs than novices and children (Brown, 1989; Gentner, 1989; Holyoak and Thagard, 1989).

Intra-domain analogies can be treated as **literal similarities** rather than as analogies (Gentner,

[10] However, using context defined in Section 3.1.2 changes the applicability of individual definitions, since the context specifies what should be compared when similarity is assessed.

1989; Vosniadou, 1989). Structural and superficial similarities are viewed as literal similarity, where superficial similarity is strictly based on surface similarities (Gentner and Forbus, 1991). The reason presented is that intra-domain analogies are similar in many simple, descriptive, and non-relational properties, in contrast to inter-domain analogies, where relational properties play a central role.

Both structural and surface similarities have comparable effects on spontaneous analogical transfer, but only the former impacts analogical transfer once a hint is provided (Holyoak and Thagard, 1989). Thus, surface similarity has a greater impact on the retrieval of a source analog than on the application of an analog (Vosniadou, 1989). However, because inter-domain analogies do not share similarity in surface attributes, they can be accessed only through similarity in their relational attributes. This also explains the difficulty in using inter-domain analogies, since people are more sensitive to similarity in descriptive properties than to similarity in structural aspects (Vosniadou, 1989). Surface similarity may enhance the probability of using analogical reminding by helping to notice that problems are of the same type (Vosniadou and Ortony, 1989). Similarity of salient attributes between the source and the target can lead to a productive analogy (Vosniadou, 1989).

Reasoning by similarity and analogy itself can make an information base more flexible, and facilitate the learning of general rules and the acquisition of new schemata (Ross, 1989; Vosniadou, 1989; Spiro et al., 1989). Thus, analogical reasoning supports the restructuring of an information base. However, reasoning by analogy may fail if the representational structures on which the analogical mechanism operates are not appropriate (Vosniadou, 1989). Although the analogical mapping may not require the presence of the underlying structure in the target domain (Vosniadou, 1989), it is required that enough be known about the target domain to make such a mapping feasible.

Summary

From the review of philosophical and psychological background on similarity presented we can conclude that a person's representation of a particular object may vary. There is also a change in reasoning if the environment changes. Thus, a flexible similarity assessment algorithm should be used in similarity-based reasoning systems. The flexible mechanism must support attention focusing and changes in the similarity assessment process.

Similarity assessment can be provided on two levels: (1) similarity among attributes (surface similarity); (2) similarity between contexts (deep similarity). Analogical similarity suggests that similarity-based systems should support the application of constraints on values, and attribute mapping. Going beyond simple examples, similarity should be computable for more than two objects. Similarity assessment should be not only a binary decision, but a continuous metric as well. Hierarchical ordering of the objects, their attributes and attribute values can be used for that purpose.

Moreover, it is possible to diminish the difference between similarity in object attributes and similarity in object relations by using a special representation language, such as Telos (Mylopoulos

et al., 1990). Here, relations and objects are treated equally.

2.2.3 Computational Background on Similarity

Similarity is central to theories of human problem solving and thus is important for artificial intelligence research. Although there are different approaches to similarity assessment, the underlying idea is to classify information according to some features, so that we can use it in similar situations. As defined in (Michalski, Carbonell and Mitchell, 1986, pp. 713), a similarity metric is: (1) a context-free mathematical measure on properties of object descriptions used in clustering – minimized for objects within a cluster and maximized for objects spanning clusters; or (2) a context-sensitive symbolic expression capturing relevant similarities between two objects – used to establish mappings in analogical inference. Usually, an artificial intelligence system assesses similarity by comparing symbolic representations of objects. The goal is to compare those parts of the knowledge structure so that the most similar cases will also be the most useful ones.

To date in computer science, little attention has been given to the theoretical foundation of similarity for computational problem solving. Rather, the focus has been on implementing psychologically plausible theories of similarity. A previously defined theory was usually tweaked to meet the system's requirements (Watanabe, 1985; Russell, 1986; Ashley, 1989; Brown, 1992; Leake, 1992b; Skalak, 1992; Michalski, 1993; Kashyap and Sheth, 1993; Sun, 1995).

Definitions

Most of the existing approaches to similarity assessment are founded on the computation of the distance between two items. The distance measure is computed either directly (i.e., the attribute value is numeric), or indirectly (i.e., the value is symbolic). The indirect distance computation is usually based on the distance in the hierarchy of terms, e.g., *is-a* hierarchy. However, computing distances from hierarchies of terms also defines semantic measures of similarity. Thus, one not only knows the strength of similarity, but has access to the semantics of it – how the objects are similar or how they can be transformed to become similar?

Since an item is usually represented by several attribute-value pairs, which may have various contributions to the solution, individual attributes are weighted. Then, an overall distance is computed either as a weighted sum (in additive models) or a weighted multiplication (in multiplicative models) of a distance between individual attributes (Nosofsky, 1990a).

Using distance measures, one could say that the apple and pear have a similarity value of 0.7. On the other hand, especially in complex and rich domains, it makes sense to measure similarity using hierarchies of attributes and hierarchies of attributes values, i.e, to consider some semantic measures of similarity. Then, for example, the apple and pear are similar, because both are edible fruit, both grow on trees, etc. Using such a similarity, we can also state the reasons for their difference, which

can be used during the adaptation process. Although strictly numeric measures of similarity are fairly limited, since they do not provide semantic information, they have another limitation, namely, systems based on distance measures often use predefined attribute weights. Thus, an apple and a pear would be equally similar in different situations. One possibility for overcoming this drawback is to use multiple (or local) contexts (Kolodner, 1993).

Similarity in Analogical Reasoning and Machine Learning

Collins and Michalski define analogical reasoning by means of several transmutations (Collins and Michalski, 1989; Michalski, 1993). They use similarity as one of the possible transmutations. They define similarity with respect to context (either implicitly or explicitly) and show how to use similarity and dissimilarity relations for inductive and deductive inference. However, they do not define features of similarity and dissimilarity, such as symmetry, transitivity and monotonicity.

Russell discusses similarity measures in analogical reasoning (Russell, 1986). After examining simple similarity approaches, Russell suggests using representation-independent similarity measures in combination with probabilistic reasoning. He uses a theory of natural kinds for features and objects. Probabilistic reasoning influences overall similarity by adding weights to individual attributes.

Veloso and Carbonell define analogical similarity as a substitution function σ, such that the similarity between literals l and l' is defined as $l = \sigma(l')$, i.e., l is similar to l' if l' can be transformed in a way that makes it equal to l (Veloso and Carbonell, 1993). They provide two versions of similarity: direct and foot-print similarity. The former is a general notion of similarity and the latter is a specific variant of the former, being tied to the goals and preconditions according to the derivational trace of the solution. Other goal-directed processes in similarity judgments are described elsewhere (Zuenkov, Kulguskin and Poletykin, 1986; Suzuki, Ohnishi and Shigemasu, 1992).

Constraints in analogical reasoning are used to ensure that one-to-many and many-to-one matches among concepts in different domains are resolved to one-to-one matches (Keane, Ledgeway and Duff, 1991). A similarity constraint is used to reduce the number of matches, by filtering out semantically different concepts. Experimental results suggest that the presence of semantic similarity makes analogical mapping easier.

Analogical reasoning relates to similarity assessment (Goldstone and Medin, 1994). Specifically, the similarity of scenes cannot be determined until the their parts are placed into correspondence. Similarly, during analogical reasoning, parts will be put into correspondence to the extent that they are consistent with other correspondences. Goldstone defines a feature match to be match-out-of-place (MOP) if the respective objects are not aligned with one another. In contrast, match-in-place (MIP) exists when features between matching objects are aligned. It has been experimentally shown that MIPs increase similarity more than MOPs.

Another approach to structure mapping is used in ARCS (Thagard et al., 1990) and MAC/FAC

(Gentner and Forbus, 1991; Forbus, Gentner and Law, 1995). Both systems use two-stage retrieval. The main idea is to retrieve all relevant cases quickly and then prune them down to retain only those of "high-quality." The second stage is used for filtering out retrieved cases based on both surface and abstract features. The biggest criticism of the MAC/FAC model is that it is too syntactic and that there is a distinction made between attributes and relations (Dierbach and Chester, 1991). Hall presents more details on analogical reasoning and compares individual computational approaches (Hall, 1989).

Similarity in Case-Based Reasoning

CBR systems use specific past experience in problem solving. When a new problem is posed, the system tries to find the most similar past problem (i.e., a case) and modify its solution so it fits the new problem. Similarity in CBR systems affects all aspects of such reasoning, namely: (1) similarity of salient features between past cases and a new case; (2) similarity of remaining features in potentially relevant cases; (3) dissimilarity of features between past cases and a new case guide the adaptation process, and may trigger knowledge acquisition;

Kibler and Aha introduced a baseline for evaluating the performance of complex similarity assessment strategies (Kibler and Aha, 1987). The simplest case-based reasoning systems use the assumption that cases can be represented by fixed set of features, and that the exhaustive search is tractable. When given a new problem, the program compares features between a new problem and all stored cases. Based on matching features, the overall similarity score is computed.

More advanced approaches have been introduced in CYRUS (Kolodner, 1983), SHRINK (Kolodner and Kolodner, 1987), and JULIA (Kolodner, 1987). Here, the similarity assessment is combined with the classification process. All similar cases are stored in the hierarchy under the same classification. If, however, the retrieved case proves not to be useful, the system makes use of a secondary classification based on failures. When a case is retrieved via such a link, it is assumed to have greater similarity to the new case being processed because of its past involvement in a similar failure.

The MEDIATOR system (Simpson, 1985) works similarly. All possible classification paths are used, retrieved cases are ranked using similarity, and a heuristic is used to prune this set further. First, cases for which the most important features are not identical, such as goals, are eliminated. Second, the remaining cases are ordered using the fixed preference scheme (background knowledge). MEDIATOR also uses a secondary classification scheme based on problem-solving failures.

The HYPO system (Ashley, 1990) is different from systems presented so far. It ranks retrieved cases as being in support of, or against an argument. In the CASEY system (Koton, 1988a) the causal model is used to infer feature equivalence and importance. The system abstracts from the actual symptoms (in medical diagnosis) to the underlying pathological states for matching.

In PROTOS (Bareiss, 1989a; Porter, Bareiss and Holte, 1989; Porter, Bareiss and Holte, 1990)

the similarity between new and recalled cases is assessed by explaining how their features provide equivalent evidence for classification. However, this approach requires domain knowledge to explain equivalence of features. The overall similarity is based on a heuristic evaluation of explanation quality and the importance of unmatched features. Because the explanation is highly dependent on domain knowledge, the system has a single context similarity assessment.

In GREBE (Branting, 1991) an explanation is used to assess similarity of a new case (GREBE works in the legal domain). A new case and precedent are believed to be similar to the degree to which explanations of key legal relationships in the precedent are applicable to the facts of the new case. Explanations are represented as cases and CBR is recursively invoked to map facts to legal relationships. Similarity assessment is in the fashion of PROTOS, i.e., single context similarity.

The FABEL project on case-based reasoning addresses the issue of similarity-based retrieval, mainly for the purpose of classification (Jantke, 1994; Voß, 1994). Thus, the cases are represented as attribute-value pairs with respect to a given finite number of attributes. Jantke argues for the need of more flexible and more expressive similarity measures. In addition, a similarity measure should allow for assessing similarity of more than two cases and should not be a local property only, but rather a global one. He also argues for the need to "abandon" the symmetry of similarity. Other pragmatic approaches to similarity assessment also abandon the symmetry of similarity based on adaptability (Leake, Kinley and Wilson, 1997; Smyth and Keane, 1996).

The AASM (Asymmetric Anisotropic Similarity Metric) algorithm reduces the case base size while preserving classification accuracy (Ricci and Avesani, 1995). As a result, the system can speed up query processing. The proposed metric uses a modified Euclidean distance to compute similarity between cases. For this purpose, a metric is attached to each case. In addition, a reinforcement learning procedure is used for adapting the local weights to the input space. Similar to the approach described in (Smyth and Keane, 1995), AASM was used to retain only highly useful cases in order to limit the case base size. However, when the accuracy was maintained in a simple domain, only a 10% case base size reduction was achieved (see Chapter 6 for more discussion).

A few issues come to attention from this summary of existing systems. These issues cover the representation and retrieval aspects of these systems. The most influential issue is that case retrieval and similarity are inter-related. Stored experience and problem-solving situations are represented in terms of surface and abstract features. These features need to be acquired from a domain expert or automatically extracted from domain knowledge. A procedure to assess similarity relies on domain knowledge in addition to case features. Domain knowledge enables inferences of featural equivalence and evaluation of the importance of unmatched features in assessing similarity. Retrieved cases are ordered using heuristics or numeric similarity functions. Usually, only featural similarities are used; systems are generally restricted to a single problem-solving context.

Although AI researchers, cognitive psychologists, and information retrieval experts take different

views on similarity assessment, there are efforts to unify them (King et al., 1988). The study shows that although similarity is highly domain dependent, some general principles can be found. Similarity is used to both guide case storage and indexing, and as a conflict-resolution method to refine or order the retrieved cases. It is arguable whether the bulk of computational effort is concentrated on representing cases or on problem solving. There are two approaches to similarity assessment: using fixed context-sensitive equivalence of features, or resolving featural equivalence during problem solving. Because there is no predefined equivalence of features in the latter approach, this approach is more flexible and thus less domain-dependent. Domain knowledge is useful for similarity assessment; however, it is not clear how to acquire such knowledge efficiently, what type and amount of knowledge is most appropriate, and how to represent it. The long-term goal of research in this area is to build generic technology that would be applicable across diverse domain.

Similarity Assessment in Databases

Jaggadish proposes an organization of objects in a spatial database, which permits efficient retrieval using shape similarity: two shapes are similar if the area where they do not match is smaller than an error margin when one shape is placed on top of the other (Jagadish, 1991). The error margin is not constant and is used to control the number of retrieved cases. This approach is similar to the contrast model (Tversky, 1977).

It is possible to resolve schematic differences among semantically related objects in multi-database systems (Kashyap and Sheth, 1993). Authors define semantic proximity to characterize the degree of semantic similarity between two objects using their real-world semantics. The key part of their definition of semantic similarity is an explicitly represented context. Thus, context in semantic similarity helps to ignore or stress certain features, using their perceived importance to the user in a given situation (Kashyap and Sheth, 1993; Peterson, 1979). Another use of their approach is to represent uncertain information and to resolve data-value incompatibility in multi-database systems.

Similarity-based reasoning is used to retrieve both exact and approximate matches for a given query (Jagadish, Mendelzon and Milo, 1995). This interest stems from the need to manage a large number of federated databases and to support more flexible and powerful retrieval.

Rafiei and Mendelzon introduced a query processing algorithm that can answer similarity queries efficiently by using the R-tree index (Rafiei and Mendelzon, 1997). They applied the system to a multidimensional data set, namely time series data. A set of transformations in a multidimensional space reflects a distance between two objects and thus is used to measure dissimilarity.

Nakamura, Sage and Iwai present a question-answering system for information bases (Nakamura, Sage and Iwai, 1983; Nakamura, Sage and Iwai, 1984). The similarity between information sources in this work is determined using a set-theoretic model of psychological similarity.

Similarity Assessment in Pattern Recognition

A pattern is usually defined as a vector of a certain dimension. Pattern recognition is based on discriminant functions or boundaries that partition the whole space into subregions representing different pattern classes (Li, Hall and Humphreys, 1993).

Watanabe introduced similarity assessment in the area of pattern recognition (Watanabe, 1985). He defines similarity as a relation between two objects. He used the principle that similar causes have similar effects. Although he realized the importance of context in similarity, his formalism was limited only to distance measures of similarity – this approach is used extensively in many other systems (Fertig and Gelertner, 1991; Pankakoski et al., 1991; Brown, 1992; Tanaka, Hattori and Sueda, 1992; Féret and Glasgow, 1993; Skalak, 1993). Because Watanabe's work is mainly oriented to pattern recognition where concepts are represented in n-dimensional space, only two measures of degree of similarity are sufficient: (1) the number of shared predicates (features), and (2) the distance between two concepts, represented in an n-dimensional space. In more complex domains, other distances between concepts should also be used, such as conceptual hierarchies (Fernandez-Chamizo et al., 1995; Girardi and Ibrahim, 1994; Lauzon and Rose, 1994; Ostertag et al., 1992; Spanoudakis and Constantopoulos, 1994), as discussed in the following section.

Watanabe also defined "The Ugly Duckling" theorem:

> *Insofar as we use a finite set of predicates that are capable of distinguishing any two objects considered, the number of predicates shared by any two such objects is constant, independent of the choice of the two objects.*

Thus, this approach to similarity assessment results in "any arbitrary two objects [being considered] equally similar". Watanabe believed in "radical nominalism", that is: "*... there is no such thing as similarity or dissimilarity in the world, as far as empirical data and their logical treatment are concerned.*" He also believed that some features are more important than others and that similarity assessment should take this into consideration. In addition, he noted that similarity depends on the purpose of classification, and classification in turn, depends on the similarity measure used. Thus, if one changes the similarity of concepts based on the purpose, then this problem is diminished.

Bar and Ullman discuss context effects on object recognition (Bar and Ullman, 1993). Psychological studies revealed that organized scenes with possible inter-object relations are recognized more accurately than unorganized scenes with impossible inter-objects relations. This phenomenon was explained by the existence of a spatial context, i.e., context that relates objects spatially. Then, partially recognizing an object's part may lead to correct recognition of other objects. Though there is similarity between recognition and retrieval of objects, the main difference is that recognition lacks the object model *a priori* and may require the object's location as well, whereas for retrieval the object model is known *a priori* and the task is to determine the object's location.

Li, Hall and Humphreys define a discrete distance space and a similarity measure for any pattern family (Li, Hall and Humphreys, 1993). The similarity measure is used to simplify the recognition process. The recognition is increased by limiting the number of selected prototype patterns.

Similarity Assessment in Software Engineering

Software engineering uses similarity assessment to support reuse. A software repository that stores artifacts, such as software code, design choices and requirements, must retrieve these components even when an imprecise query is posed. In addition, if the perfect match is not available, similar components should be recalled. The main problem, is to choose an appropriate representation of software artifacts and deciding on a suitable similarity measure.

A software repository is a special-purpose information base, intended to store reusable software entities. The granularity and form of entities suitable for reuse can vary: it can be at the requirements, design or code level. A successful repository must support efficient and flexible retrieval of repository items. Efficiency means that the repository retrieval system will scale up as more entities are acquired. Flexibility means that the criteria for retrieval should not be predefined, since they may change over time or they may differ among individual users. Moreover, the system should retrieve useful entities even if the query is not specified completely and should retrieve similar entities if the exact match to the query is not possible. There should also be an option for recall-oriented retrieval (all relevant entities should be retrieved, even if some irrelevant ones are included) and for precision-oriented retrieval (only relevant entities must be retrieved).

Software components can be located by library cataloging, database classification (Daudjee and Toptsis, 1994), query-based retrieval (Tedjini et al., 1990), keyword- and attribute-based retrieval (Diab, 1992), formal specification (Rittri, 1991; Mittermeir, Mili and Mili, 1993), information retrieval schemes (Díaz and Freeman, 1987), hypertext-based classification and retrieval (Garg and Scacchi, 1989), similarity-based organization (Schwanke, 1991; Adams, 1993), analogy-based retrieval (Lee and Harandi, 1993; Jeng and Cheng, 1993), case-based reasoning (Fouqué and Matwin, 1993; Fernandez-Chamizo et al., 1995) and hybrid organizations (Constantopoulos et al., 1994).

The most appealing feature of similarity-based retrieval is that it recalls relevant information even when the user cannot specify the query completely and precisely, or when exact match does not exist. It also supports iterative browsing, which is needed when a repository is large, and created and used by many users. In general, it is impossible to predict what needs to be retrieved, when, and in which form. Thus, the similarity measure should use some form of contextual information to define various preferences and should allow for iterative modifications of the query.

AIRS (AI-base Reuse System) (Ostertag et al., 1992) is an AI-based library system for software reuse. AIRS is a hybrid system that integrates faceted indexing and a hierarchical frame system. Similarity between two components is based on a distance measure of their respective descriptions.

The distance measure is based either on the closeness or the subsumptions relation. The closeness relation is intended to capture the fact that new components can be constructed by modifying existing constructs. The subsumption relation is defined among the components of the system and is intended to capture the idea that certain components can be built by combining several other components together. On the basis of this measure, two components are considered similar if the total difference between their corresponding terms is small. The main restriction of the system is its inflexibility. Because the components must be described in terms of all features, adding new features would involve modifying all components that are already stored in the library.

The CAReT system (Lee and Harandi, 1993) is an analogy-based retrieval system, applied to software design reuse. It supports incomplete query specification as well as similar and exact matching of components. Its information base consists of background knowledge and a design library. The system uses *is-a* and *part-of* links to navigate through the library. The retrieval mechanism uses a three-phase procedure: locate, evaluate and select. Flexible retrieval is supported by modifying the level at which attribute matches occur. The system uses two levels of matching: object- and attribute-based. The first level compares object classes, object types, attribute properties and their relationships to other object types. The second level compares attribute data types and properties.

The SIB system (Software Information Base) uses a distance-based algorithm to measure similarity between software artifacts (Constantopoulos et al., 1994; Spanoudakis and Constantopoulos, 1994). The model is based on the compound distance measure, consisting of the following: the identity distance (distinguishes non-identical and identical objects), the classification and the generalization distances (measures the depth of a class in the *is-a* hierarchy) and the attribution distance (expresses the distance between the attribution of two objects).

The Reuse Assistant approaches the problem of submitting a query by combining statistical methods with case-based reasoning (Fernandez-Chamizo et al., 1995). Incremental query construction is used and the relevance of retrieved items is determined statistically. The system was designed to be used for retrieval of object-oriented programming classes. The user submits a free form query, which is processed using information retrieval techniques. In addition, the system supports browsing through the repository. After the initial free form is processed, the system returns a frame with slots to be filled. This stage of retrieval is more similar to an approach used in case-based reasoning.

Similarity Assessment in Other Areas

Similarity measures are important in computational systems. Conceptual clustering uses a similarity measure to properly cluster objects. If no similarity measure is used then we get an arbitrary clustering (arbitrary domain structure). It is well recognized in this area that defining a similarity measure is not an easy task. A straightforward approach is to compare the number of matching attributes to the total number of attributes defined for compared objects (Tversky, 1977). It was

shown that given all observed and derived attributes, this approach renders any object equally similar to any other (Watanabe, 1985). This principle is known as "Ugly Duckling Effect".

A similarity measure is used in conjunction with conceptual graphs to define concepts by means of a series of examples (Maher, 1993). The semantic distance between two concepts is defined as the shortest path between these concepts in a type hierarchy. The similarity measure is then represented as an interval, defined by the necessary support (lower bound) and the possible support (the upper bound), as obtained for each graph and taken as a percentage of the whole range.

Green uses semantic similarity to construct hypertext links (Green, 1997). Semantic links between paragraphs and articles are generated using lexical chains, which are sequences of semantically related words in text. The semantic information may be obtained from any lexical resource.

In various studies a correlation between approximate reasoning, fuzzy reasoning and similarity assessment has been established (Binaghi et al., 1993; Ruspini, 1990; Ruspini, 1991). Ruspini defines a similarity relation as a fuzzy relation that is reflexive, symmetric and transitive. Thus, the notion of a similarity fuzzy relation extends the notion of an equivalence relation. In Binaghi et al., similarity classes are defined as fuzzy sets in the universe of discourse of fuzzy labels. The number and the kind of similarity classes vary, depending on the context. Here, fuzzy reasoning is used for image analysis and the relevant features for comparison may change based on the task.

Similarity assessment and uncertainty reasoning are related. The former measures the distance (either numerical or semantic) between objects. The latter uses a certainty factor to estimate the quality of the result of the reasoning, i.e., to measure the proximity of the answer to the desired result. Thus, the statement in uncertainty reasoning about a value of an attribute being in the high certainty range can be expressed using similarity reasoning terms as follows: all objects with a similar values for all selected attributes can be treated as equivalent. In CBR a similarity measure is used to locate and order relevant cases. The number of retrieved cases depends on the retrieval strategy and in some systems it can be dynamically controlled by relaxing or constraining the initial query (Gaasterland, 1993; Lauzon and Rose, 1994; Rissland et al., 1993).

2.3 Discussion

The chapter presented a literature review of case-based reasoning. Similarity assessment was examined from a philosophical, psychological and computational points of view. Representative approaches were compared on the basis of their foundation and their suitability for certain tasks. The review shows that psychological approaches are concerned with similarity assessment that is consistent with human cognitive models. In contrast, computational approaches focus on the utility of the algorithms proposed, i.e., on their competence and scalability. Whether they share a cognitive model with human similarity assessment is of secondary importance.

Similarity assessment is important for the success of CBR systems, but current approaches do not address flexibility and scalability issues. Thus, in the next chapter we propose an alternative representation of cases and a novel similarity assessment theory. In formulating the proposal, we have followed a "performance approach", i.e., we define similarity assessment algorithm that improves both competence (i.e., quality of the reasoning process) and scalability of CBR systems. Our work on similarity was motivated by the following requirements:

- The need for a knowledge representation language that diminishes differences between attributes and relations and thus reduces the difference between surface and deep similarities. The representation should support properties of similarity such as monotonicity, symmetry, transitivity.
- The need for a hierarchical concept representation (objects, attributes, values) to support semantic equivalences and similarity in addition to numerical distance measures.
- The need for an explicit definition of context to support attention focusing. This also diminishes the "Ugly Duckling Effect" and enables goal-dependent similarity judgments.

As will be shown in Chapter 6, this novel similarity assessment theory supports efficient implementation of an incremental retrieval algorithm, and flexible computation by trading off the precision of computation for the resources needed.

Chapter 3

Flexible Similarity Assessment

This chapter proposes a similarity assessment theory that uses an explicitly defined context. Our motivation for this work is the lack of flexibility and expressiveness of previous approaches to case retrieval, and the lack of psychological evidence for measuring similarity numerically.

Based on the proposed theory of variable-context similarity assessment, we define a case retrieval algorithm, used for retrieving relevant cases efficiently, even if task or user preferences change.

First, we identify what it means for a case to be relevant in a given situation. Webster's dictionary defines *relevance* as: (1) Pertinence to the matter at hand, and (2) the capability of an information retrieval system to select and retrieve data appropriate to a user's needs.

For case retrieval we adopt the second definition. In database systems, a user specifies a query and the system retrieves everything that matches the given description. Thus, everything retrieved is relevant to the given query. In CBR systems, a query corresponds to a partially described case without a solution. Another distinctive feature of CBR systems is that even if the problem is described completely, usually there is no exact match in a case base. Hence, the system must support approximate retrieval, i.e., recall partial matches along with exact ones. To ensure that only relevant cases are retrieved, a similarity assessment algorithm returns partially ordered cases based on how close they match the query. Informally, *a retrieved case is considered relevant if it is similar to the query.* Particular details on how similarity matching is performed are provided in later sections.

Several similarity assessment algorithms have been proposed, yet we believe that they are inflexible for case-based or analogical reasoning. The reasons for this can be summarized as follows:

- Numerical similarity measures based on distance metrics are ill-suited for complex domains where a hierarchy of attributes and their values need to be considered.
- Pre-defined context (i.e., a definition of how case similarity is measured) makes it hard to

change the similarity result when task or user preferences are modified.

- A predefined retrieval approach (either recall- or precision-oriented retrieval) is difficult to modify based on changing needs.
- Pre-defined attribute importance provides good results only for a specific task and user.

This chapter proposes a novel theory of similarity based on the notion of context, where similarity is computed using a symbolic measure and conceptual modeling. This theory makes it possible to change the definition of similarity for entities depending on the task under consideration and user preferences. The proposed similarity assessment theory also supports an efficient and scalable implementation of the similarity assessment algorithm. In addition, similarity assessment can be flexibly controlled by trading off accuracy, precision and recall for computing resources. In other words, variable-context similarity assessment produces distant partial matches faster, with a possibility to iteratively remove less relevant cases.

One of the advantages of CBR systems over rule-based systems (RBSs) is that they do not compile knowledge during the acquisition process. This may result in a slower execution, it will require larger storage space, but it is a more flexible approach that can be applied even in less understood domains. While RBSs use knowledge in an already transformed form – as general rules, CBR systems store experience – the problem, solution and feedback. Thus, CBR systems delay knowledge processing until the reasoning process is started. At this time, new relevant facts may be available. This also solves the problem of over-generalization of experience, and handling exceptions to general knowledge. In addition, the specific situation or the user's needs may change. Thus, CBR system are based on a more flexible reasoning paradigm than RBSs.

Most of the existing CBR systems rely on indexing as a way of recalling relevant cases. This can be put into analogy with the problem of rule-based systems, where rules need to be compiled prior to using them – in CBR systems, indexes are created prior to using the knowledge. Indexing increases the efficiency of retrieval by hard-coding the relevance measure. The problem with relying on predefined indexes when accessing relevant cases is that it diminishes the advantage of CBR systems over rule-based systems, since part of the knowledge is hard-coded into the system. We believe that, even though cases should be structured to allow for efficient retrieval, the criteria for retrieval should not be predefined. The reason for this belief comes from the above mentioned flexibility problems.

Later we present a theory of variable-context similarity assessment. This replaces the standard indexing approach in order to retrieve cases flexibly. The retrieval process is based on biasing the search towards relevant cases using an explicitly defined context, which may be modified during this process.

3.1 Motivation for Variable-Context Similarity Assessment

We may use "information bases" as a neutral term which subsumes databases, knowledge bases, case bases, software repositories, etc. We assume that items in an information base are represented as attribute-value pairs. The remaining of the chapter deals with similarity-based retrieval, which uses explicitly defined context.

Intuitively, relevant information is the one which is useful. If we know what is relevant, we may retrieve appropriate information by posing a specific query. A query in a CBR system is a description of a current problem (a partially-developed case without a solution and feedback). Most of the time, there is no exact match between the source case (current problem) and cases in a case base. The goal is to produce a relevant answer even if the query is not completely specified. This can be handled by the use of a similarity-based retrieval algorithm.

Similarity-based retrieval is used for assessing similarity between a source case in a case base (i.e., cases with the problem, solution and feedback) and a target case in the query (i.e., a case without a solution and feedback). A query specifies the input case (possibly not completely) and the criteria for matching (i.e., minimum and maximum number of cases desired to be recalled, possibility to automatically relax or restrict the query, etc.). The system retrieves all similar cases.

Psychological experiments and everyday experience show that what is considered similar in one situation may not be similar in another. Thus, systems that do not take context into consideration may return correct answers in some situations, and unreasonable responses in other situations. To prevent this, the proposed similarity assessment theory represents constraints on similarity matching (context) explicitly. Using context we can specify which parts of the case representation to compare and what kind of matching criteria to use. Context helps to retrieve relevant cases and to exclude similar but irrelevant ones. Thus, context can be used as a basis for relevancy measure, i.e., cases are considered relevant if they are similar with respect to a given context.

Recall that similarity between cases may change with respect to users' needs, while solving different tasks. Thus, a flexible similarity-based retrieval system must not have predefined notions of similarity. To accomplish this we define a variable-context similarity assessment, where context is a set of attributes with associated constraints on the attribute values. Context specifies how the matching should be performed, giving us local matching criteria (Kolodner, 1993), and the relevance measure. Context is also used for filtering out all irrelevant pieces of information and can be applied in multi-functional information bases (Mylopoulos and Motschnig-Pitrik, 1995).

3.1.1 Tasks to be Addressed by Variable-Context Similarity Assessment

The most basic task in similarity assessment is to determine if particular cases are similar. This may include comparing two cases or comparing a set of cases. We show that context plays a crucial role

in this process and we define a satisfiability function. The case retrieval task involves the retrieval of all cases that are similar according to specified criteria. We show how context enables us to specify matching criteria flexibly. We also define a retrieval function (see Section 3.5.2). This function addresses a thesis goal – to flexibly and efficiently retrieve relevant cases.

Knowledge mining and case base organization tasks are used to help locate important case descriptors and to organize case base into meaningful clusters of cases that improve both system competence and efficiency. We show mechanisms for finding attributes relevant for a particular task. These attributes represent context, and for a given set of cases they represent the most restrictive context. The most restrictive context is characterized by maximizing the number of attributes used and minimizing the number of constraints applied to attribute values. Using multiple possible contexts, a given case base can be organized into context-dependent clusters of cases that are relevant during solving the same task under similar conditions. The function for finding representative contexts is called explain function (see Section 3.5.3), since it explains why a particular descriptor is relevant and which cases should be kept related.

The main goal of our research is to define a theory for flexible and efficient similarity-based retrieval and for context-based knowledge organization. First, we define case base, case, context, context transformations, satisfiability, equivalence and similarity functions. In addition, in Section 3.6 we study features of context-based similarity in retrieval, namely reflexivity, monotonicity, symmetry and transitivity.

Flexible Similarity Assessment

Another way to classify similarity assessment approaches is to compare their implementation as algorithms. There are two possible approaches: the first relies on predefined similarity relations among cases and the second locates similar cases at query time.

If similarity relations among cases are predefined, the flexibility of the system is limited and the system can work properly only in one context. The reason is that the conditions under which the cases are compared, are predefined or only a limited flexibility is allowed. A similar approach where all cases are retrieved and ranked based on a global matching strategy appears in (Kolodner, 1993).

If similar cases are located by defining similarity relations at query time, then the conditions for matching can be flexibly changed. We refer to conditions for matching as context, and we define similarity between cases with respect to the context. In general, similarity is a relation with three parameters: a set of relevant cases, a context (matching conditions), and a case base. Similarity during retrieval is used when the context is specified and the task is to retrieve all relevant cases from a case base; thus, a case base and a context form an input and a set of retrieved cases is an output. This approach is similar to local matching criteria (Kolodner, 1993) since several possible strategies can be used for matching. The difference is that most previous approaches to local matching use

numerical measures of similarity and thus a semantic explanation of similarity may not be available.

The first approach to similarity-based retrieval is obviously not flexible but can be used to answer the question "are the items similar?" and not the question "how are they similar?" (e.g., in which context are they similar). Because the similarity relation among cases is embedded into the system, context is static and thus the system provides a sound inference only in these "built-in" situations.

The second approach is more favorable because it supports changing context, and gives us additional semantic information about compared items. Thus, the following questions can be answered: "how are the items similar" or "in which context are the items similar." Flexibility and additional semantic information are crucial for reasoning, since what is relevant in one situation may not be relevant in another. Thus, we need flexible similarity assessment and we need to know on what grounds the items are similar. Knowing how the items are similar and exploiting the monotonicity of similarity with context enables automatic control over the number of retrieved cases.

A central problem in similarity-based retrieval systems is assessing the relevance of the retrieved information to the current goal, specified through the query. This problem becomes more difficult to solve when (automatic) iterative retrieval tools are used: When the initial query does not yield a satisfactory result, it may be modified (fine-tuned) by either the user or the system. This modification may involve the strengthening or weakening matching criteria. Such operations change the direct relation of the initial query and the relevance of retrieved items. Thus, the relevance of items retrieved using the modified query needs to be assessed. Similarity theories developed so far (Tversky, 1977; Gick and Holyoak, 1980; Gentner, 1983; Holyoak, 1985; Vosniadou and Ortony, 1989; Leake, 1992b; Skalak, 1992; Suzuki, Ohnishi and Shigemasu, 1992; Michalski, 1993) cannot be used during flexible relevance assessment of the system's response. Though they enable partial matches, the initial query (a case) is not modified and thus there is no support for iterative browsing or for explanation of the relevance of retrieved items.

3.1.2 The Role of Context

In various research areas, context is defined differently and sometimes different terminology is used. Context in natural language understanding helps specify a word's semantics (disambiguation). It represents a situation, a portion of the world that is relevant to the determination of the content of the sentence (Barwise, 1989; Cardie, 1993a). According to the Collins Concise English Dictionary, context is defined as *"the parts of a piece of writing, speech, etc. that precede and follow a word or phrase and contribute to its full meaning"*.

In databases, the term "view" is used to present a database object in relation to a particular situation, e.g., to present partial and consistent viewpoints of the contents of a database to different users (Ceri and Widom, 1991; Bertino, 1992; Motschnig-Pitrik, 1995). Thus, a view can be described as the context in which database objects are seen. In knowledge base management systems, context

is used to partition information bases (Mylopoulos and Motschnig-Pitrik, 1995). For this purpose, contexts are defined as special units (basic building blocks of knowledge bases), comprising contents (units included in a given context), lexicon (the local names) and authorization rules (rules guiding transaction management for individual users).

In pattern recognition, contextual knowledge is known as an "aspect", i.e., a certain characteristic of an object (Watanabe, 1985). Partial recognition of an object's part within a configuration of several objects can be used during recognition of other objects (Bar and Ullman, 1993). This is referred to as a spatial context.

Context in human-computer interaction is defined as *the environment of communication, which enables the intended meaning to be ascribed by the recipient of the data* (Wixon, Holtzblatt and Knox, 1990; Edmondson and Meech, 1994). In the hypertext-based explanation system VICE (Huuskonen and Korteniemi, 1992), context is defined as an arbitrary combination of dimensions (object, attributes, values, task, component, abstraction). In neural networks (or in other perceptual systems) the context is used implicitly - the system performs specific organizational decisions based on knowledge it has acquired about the data.

In model-based design, three types of context are defined: the *structural context* of a component consists of its physical properties and the components to which it is structurally related, the *behavioral context* of a component consists of its behavior and the behavior of related components and the *expected behaviors* are abstract descriptions of device behavior (Nayak, Joskowicz and Addanki, 1990). In propositional logic a context is modeled as a set of truth assignments that describe possible states of affairs of that context (McCarthy, 1993; Buvač, Buvač and Mason, 1994). In (Attardi and Simi, 1993) viewpoints are considered as a formalization of the notion of context. Contexts serve as a frame of reference and rules are provided to relate one context to another.

The motivation for using context is its ability to bring additional knowledge to the reasoning process and thus focus attention on relevant details (Light and Butterworth, 1993). When retrieving cases, we might want to consider all attributes used to describe them or only certain subsets of them (in different situations different subsets may be relevant).

As was shown in (Murphy and Medin, 1985), attention is important in categorization. It was observed that when people categorize objects or reason about them, they often shift attention among sets of perceptual or deep features. People's explicit causal beliefs about objects guide the selection of features used. These changes can be modeled using an explicitly represented context.

The notion of context in analogical reasoning was presented in (Collins and Burstein, 1989) as "an important test of any of these (analogical) computer models":

> *[An important test] is whether they can select two different correspondence mappings from a source domain depending on what aspects of the source domain are relevant to the target domain. None of the models has, as yet, addressed this central problem directly.*

Everyday experience also suggests that context brings additional information to the reasoning process. Although the role of context is accepted by the research community, most of the current systems for determining similarity use only a predefined context (or a closed set of contexts).

As a simple example consider an information base consisting of personnel records. Let us suppose that the information base is used by a job-placement agency. It is relatively straightforward to specify what kind of features should be represented in the information base. However, it is also apparent that if one wants to find a person for a programming job in a prestigious company, important features for retrieval of relevant information (cases that correspond to perspective job candidates) would be skills such as: experience in programming, ability to work in a team, etc. Physical appearance is usually not an issue in such a context. If a new model is needed for a local fashion store, however, then the most important feature might be physical appearance (more often than not, programming skills of the candidate would not count as a plus). Thus, even though in both cases the general task is the same – to find a person right for a job – different criteria are applied in order to retrieve the desired information. In other words, only certain parts of the representation are used during individual retrievals. Context is used as a mechanism to filter out all irrelevant pieces of information and to specify the criteria for matching.

Similarity judgments are made with respect to the representation of entities, not with respect to the entities themselves (Medin and Ortony, 1989). Thus, having a changeable representation, i.e., supporting flexible addition and deletion of attributes from an object representation, implies that one can make any two cases similar according to some criteria. To control this process, a context may be used as a way of focusing the similarity assessment only on certain features of the representation.

Context can also be used to exclude irrelevant information. This feature is particularly important in unusual situations where similarity may be affected by a "usual" (or prototypical) context, e.g., a context in which the cases appears most of the time.

Context Dependency

Tversky (Tversky, 1977) discussed the dependency of similarity on context and frame of reference; however, his formalism did not capture this dependency. Tversky did not consider changed or implicitly assumed context as a phenomenon affecting similarity judgment.

Another illustration of context playing a central role in determining similarity is the example presented in (Tversky and Gati, 1982). Here, faces with fewer features are rated as more similar than faces with many features. One explanation of this experimentally-proven phenomenon is that in the former case subjects considered all features to be the context, whereas in the latter case there was no specification of the context and one could not use all features to determine similarity.

In (Barsalou, 1989, p. 77), a further example supporting the context dependency of similarity assessment is presented, where the instability of similarity judgments is explained. It is found

that "something heavy" is a better cue than "something with a nice sound" for retrieving the sentence "The man lifted the piano." In contrast, "something with a nice sound" is a better cue than "something heavy" for retrieving the sentence "The man tuned the piano." This shows how different attributes of the concept piano are important for different tasks. Barsalou believes that certain knowledge about a category – context-dependent information – becomes active only if relevant in the current context. Moreover, it is believed that category representations vary across contexts and that the properties composing these representations also vary. We agree that the similarity judgment should be dynamic, and only a relevant portion of the representation should be considered in a particular context. However, the knowledge representation scheme should be rich enough so the objects can be represented in different contexts.

It would be safer to consider the "context-independent" information as information which automatically activates a specific context for a certain object each time this object is accessed, regardless of the current context. Thus, "context-dependent" information would not activate a specific context and would be dependent on the current context.

The necessity to distinguish between context-independent and context-dependent information, can be shown by considering the saying: *One cannot compare apples to oranges.* This statement is typically used in everyday life with the meaning that two things are incomparable (because they are different). However, it is a false statement in general, unless one assumes an implicit context. If we want to compare tables, apples, oranges and cars with respect to edibility, apples and oranges are not only comparable, but also quite similar in this context. Similar ideas are described by Saaty, who states that "we can compare apples and oranges by decomposing our preferences into the many properties that apples and oranges have, ..." (Saaty, 1996).

Certain information automatically activates a specific context for a certain object each time this object is accessed, which is the result of stimulus bias (Barsalou, 1989; Nosofsky, 1990b). For example, the word "skunk" most often evokes the notion of an unpleasant smell regardless of the current context, whatever it might be. This is possible, since the particular attribute has high "accessibility". Note also that the stimulus bias varies between individuals and can also change over time. This is why similarity assessment needs to be flexible and why it is not necessarily sufficient to have predefined similarity relations.

Context-dependent information can either be represented with the object or inferred. In the latter case, when reading about zebras one may infer that zebras have ears from the knowledge that zebras are mammals and all mammals have ears.

Besides using context-dependent and context-independent information, Barsalou (Barsalou, 1989) also employs *recent context-dependent information.* According to psychological studies, context-dependent information can become context-independent once activated, but without subsequent reinforcement it will lose this status. It is proposed that the similarity of two objects depends

on the amount of overlap among their context-independent, context-dependent and recent context-dependent information. The relative weighting of the information may vary based on contexts and thus experience can affect information dependency on context. This is important, since it supports our claim that CBR systems have an advantage over RBSs. In addition, it motivates for CBR systems not to use predefined indexing schemas.

Assume the comparison of a child to its parents. Typically it is odd to say that the parents are similar to the child. The reason is that usually we know parents earlier than children. However, despite the natural generalization hierarchy, if we know children earlier than parents, an odd comparison would seem to be to relate parents to children. Again, the reason is that one assumes an implicit context. If the context is stated explicitly, one can compare the similarity of parents to children as well – for example, we can state that a son is similar to his father because both have brown eyes and black hair.

Context and Relevance

As mentioned earlier, context supports flexible information retrieval. Retrieval of useful information is accomplished by focusing attention on relevant parts of knowledge or by locating both exact matches and similar cases.

There is little commonality between definitions of relevance used in different systems (Greiner, 1994). The most common ground is reached when relevance is defined as something useful. Our definition of relevance can informally be stated as follows: *A retrieved case is relevant if it is similar to the query with respect to the current context.* Thus, context and similarity are used to determine a relevance measure.

Another motivation for defining relevance with respect to the context is the dependency of context on situation and required view (Acker and Porter, 1992; Kashyap and Sheth, 1993). The notion of a context is also used in artificial intelligence in order to find information that is relevant in a given situation. Using a finite representation of an item (a realistic assumption), a changing context may affect the similarity results (see Figure 3.1). A query specifies the context, and the goal of retrieval is to find all relevant items in a case base, i.e., all cases which are similar to the query for the specified context. On the one hand, if the context is defined as the *area*, then items a, e, and d are relevant (see Figure 3.1, Example 1). On the other hand, if the context is changed to *color* : *dark* then items c and d are considered relevant (see Figure 3.1, Example 2). Even though this is an artificial example, similar situations occur in real life. The criteria for similarity measure often change with differing tasks or reasoning situations.

Using the example in Figure 3.1, we argue that systems with predefined similarity, i.e., systems with implicitly-stated context, work properly only in situations covered by such context. Even though this inflexibility is not problematic in domains where the context stays unchanged during

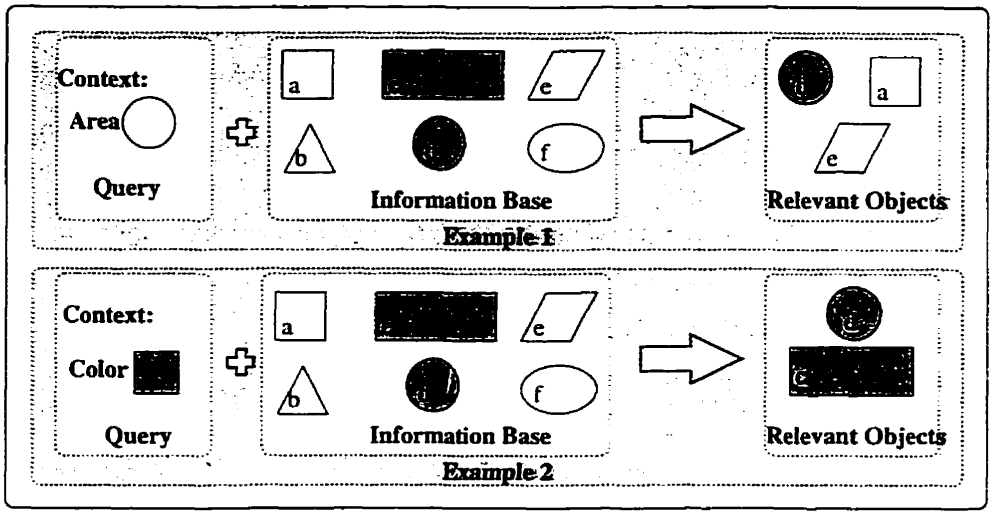


Figure 3.1: *Two examples of how context affects similarity.*

the reasoning process, it is a limiting factor for applications where context may change because a user's needs change as the information base evolves (see Sections 6.2.1 and 6.2.6 for examples).

Harmful Context

Even though we presented several reasons why and when context is helpful in problem solving, there are also situations when the use of context may result in a misperception. Davies and Thomson observed through several psychological that when a new person was introduced to participating subjects in a previously seen context, the amount of false recognition was increased (Davies and Thomson, 1988, p. 294-5). Another interesting observation revealed that adults are less faulty in such recognition (33% wrong) than youths (50% wrong), which can be explained by the higher amount of knowledge possessed by adults (Davies and Thomson, 1988). These experiments may lead to the conclusion that: (1) context is implicated in the mechanism which generates false recognition; (2) the interactive effects of context and retention interval operate differently for false recognition and correct recognition; and (3) the effect of context on false recognition over time seems to depend on the type of context, i.e., it depends on if the context is applied to specific events (episodic memory) or to abstract knowledge (semantic memory).

Since context can have a harmful effect on the reasoning process in some situations, it might be necessary to apply a decontextualization process (Davies and Thomson, 1988). The need for this is apparent from the existence of episodic memory that is based upon memories of a specific experience. In contrast to this, semantic memory consists primarily of abstract knowledge.

Decontextualization derives abstract knowledge from the specific experience. Knowledge is changed from contextually- to semantically-referenced knowledge. This process may also be referred to as generalization, since it extends learning to situations beyond the original learning context. The

major problem here is the fact that episodic memory depends upon environmental context; thus, decontextualization must overcome these obstacles. McCarthy and Buvač (McCarthy and Buvač, 1994) consider decontextualization as a process of transforming several contexts occurring in the discussion into one context which is their generalization. However, generalization may create analogues – a problem's isomorphs. The utilization of such knowledge is not straightforward since a context-free object has a limited value if the utilization is not context-free (Lockhart, 1988).

As will be shown in the next chapter, this harmful effect of context is due to the fact that if the context is not defined explicitly, one usually assumes an implicit context. Thus, if the actual context changes from the usual one, the reasoning process may be invalidated. This is also the reason why, for example, an IQ test would not find a genius since it tests only for average people, i.e., people who follow the standard implicit context in their reasoning and problem solving.

Global and Local Contexts

In order to define and represent context, it may be useful to distinguish between local and global contexts. The former will have smaller scope than the latter. In addition, the global context defines local context.

We propose that a global context consists of the task description and goals. A local context specifies fine-grained contextual knowledge to be used during the retrieval of actual cases in order to satisfy the goals of the global context. In subsequent chapters we describe how the retrieval task specifies goals and how global context specifies both the criteria for retrieval as well as the context.

As an example of global and local contexts, assume an assembly line robot. The task could be described as "assembly of a particular part of a car." This may imply the goals – "move the end-point from X, Y, Z to $X'Y'Z'$ pick up part B; move to point X'', Y''', Z'' and affix part B into part A''. Each individual goal needs to be satisfied for the whole task to be completed. A CBR system could specify criteria and local contexts in order to retrieve supporting cases to satisfy the goals (note that one case may not solve all goals). Later we show how a user can interactively affect the retrieval process.

3.2 Knowledge Representation

Because representation and retrieval are inter-related, different similarity assessment strategies require different representations. More complex reasoning in complex domains can be achieved with a hierarchical representation. This may involve hierarchies of cases, attributes, and attribute values. Variable-context similarity assessment is used for retrieving close as well as exact matches. We define context and use it to control the degree of similarity among individual cases by explicitly stating criteria for matching. This section introduces definitions on which the novel theory of similarity

assessment is founded (Jurisica, 1994; Jurisica and Glasgow, 1996a).

A representation also affects the similarity function and thus it also affects features of similarity such as reflexivity, symmetry, transitivity and monotonicity. In many studies and systems only asymmetric similarity is supported (Tversky, 1977; Smith, 1989; Gentner and Bowdle, 1994; Ohnishi, Suzuki and Shigemasu, 1994; Jantke, 1994; Ricci and Avesani, 1995). However, different representations enable symmetric similarity (Carnap, 1967; Ruspini, 1991). As is evident from the review presented, because similarity is not an objectively existing measure, we cannot assess features of similarity for all possible applications. Instead, each individual application should be designed on the basis of what properties of similarity (or similarity features) are desired, and the designer should choose an appropriate representation formalism.

Many of the systems presented use explicit similarity assessment because they operate on weak domain theories. In tractable and perfect domains the similarity can be dynamically computed if an appropriate knowledge representation scheme is used (Porter, 1989; Thagard and Holyoak, 1989; Lauzon and Rose, 1994; Rissland et al., 1993). The advantage of dynamically computed similarity, compared to a "hard-wired", is that the system can assess similarity during problem solving while taking into consideration different contexts (or views) and different tasks or user preferences. This approach supports adapting a similarity measure which is a necessity in a dynamic environment.

3.2.1 Representation of Cases

Generally, a case represents a real-world experience as a finite set of descriptors, which define the problem, its solution and feedback. Each descriptor comprises pairs of attribute names and their values. Individual attributes are grouped into categories. Each case in a case base has at least two categories: one comprising attributes that describe the problem, and the other that groups attributes describing a solution. Later we will show, that grouping individual attributes into categories may increase system competence. We will also discuss methods that can be used to properly group attributes into categories.

Definition 3.2.1 (A case) *A case, C will be represented as a finite set of unique attribute-value pairs:*

$$C = \{\langle a_0 : V_0\rangle, \langle a_1 : V_1\rangle, \ldots, \langle a_n : V_n\rangle\} = \{\langle a_i : V_i\rangle\}_0^n = \{A_i\}_0^n,$$

where $A_i = \langle a_i : V_i\rangle$ is an attribute-value pair.

Associated with every attribute a_k is the attribute domain D_k, consisting of the possible values the attribute can take. All values for a given attribute must belong to its domain: $V_k \in D_k$. Values can be complex as well as null or "not available." It should be noted that multiple null values are allowed, each for a given domain, which is similar to Imielinski and Lipski's (1984) approach. This supports a more flexible use of null values during matching as will be described later.

Using information about the usefulness of individual attributes and their properties,[1] attributes are grouped into one or more categories. Categories bring additional structure to a case representation. This reduces the impact of irrelevant attributes on system performance by selectively using individual categories during matching (Aha, 1992a; Ortega, 1995).

A *source case* comes from the case base, i.e., it has a solution and feedback. An *input case*, also referred to as a *problem* or *target* case, describes a given problem, either fully or partially, and it does not have a solution or feedback.

Definition 3.2.2 (Case base) *A case base, $\Delta = \{C_1, C_2, ..., C_k\}$, is a finite set of cases - the search space of source cases. A case base is an information base for a CBR system.*

During the reasoning process cases are compared on the level of attributes and attribute values. Clearly, values can be compared only after respective attributes match directly (i.e., they have the same attribute name) or if they match using an analogical mapping (i.e., attribute names are put into equivalence using mapping). There are four possible outcomes of attribute matching: sets of attributes of individual cases are disjoint, they are equal, one is a subset of the other, or they share a common subset of attributes. If compared cases agree on all attributes and values, then we call the match an *equality*. Two cases *partially match* if they are neither equal nor disjoint, i.e., they agree on a subset of attribute-value pairs. The degree to which one case partially matches the other depends on how many attribute-value pairs do match.

The matching process can be controlled by grouping attributes into categories. Different matching criteria may be posed on different groups of attributes. Next we show how context can be used to control the degree and the form of the partial match.

3.2.2 Representation of Context

In short, context is a parameter of similarity that can be used to map a case base onto a set of relevant (in terms of context) cases. Context controls what can and what cannot be considered as a partial match. In other words, context specifies what attributes are involved in similarity assessment between cases, and what set of values may be considered for these attributes. It defines which aspects of a case are important in a particular situation.

Definition 3.2.3 (Context) *A context, Ω, is defined as a finite set of attributes with associated constraints on the attribute values:*

$$\Omega = \{\langle a_0 : CV_0\rangle, \ldots, \langle a_k : CV_k\rangle\} = \{\langle a_i : CV_i\rangle\}_0^k,$$

[1] This information is obtained either from domain knowledge or with help of a knowledge mining algorithm, such as `explain` function described in Section 3.5.3.

where a_i is an attribute name, and CV_i is a representation of a set of possible values for a_i, i.e., if D_i is the domain for a_i then $CV_i \subseteq D_i$.

In the proposed theory, a constraint set CV_i for an attribute a_i with domain D_i can be represented in a number of ways:

1. as a finite set of allowable values, i.e., $CV_i = \{V_1, V_2, ..., V_n\}$;
2. as a collection of ranges of allowable values, i.e., $CV_i = \{x : x \in D_i \wedge lowerLimitValue \leq x \wedge x \leq upperLimitValue\}$;
3. by performing union, i.e., $CV_i = D_i \cup S$, performing intersection, i.e., $CV_i = D_i \cap S$, specifying prohibited values, i.e., $CV_i = D_i - S$, where S is some set of values specified using representations of type 1, 2 or 3.

A context can be used to view (Acker and Porter, 1992; Kashyap and Sheth, 1993) cases in terms of a given problem description. In other words, a case is interpreted with respect to a given context. In the proposed theory, the relevance of a particular case to a given context is based on attempting to match the value of the case attribute with the corresponding constraint set in the context. Depending on the importance of the attribute, this can be carried out in three possible ways:

- Totally ignore the attribute. Such an attribute would not be included in the context, thus it would not matter whether or not the attribute is contained in the case. As an example, category solution is usually ignored during case matching.
- Consider only the existence of the attribute, but not its value. For such an attribute a_i, we would specify a constraint $CV_i = D_i$, thus if the attribute is present in the case it would automatically satisfy the constraint.
- Use both the attribute and its value. Thus, if $\langle a_i, CV_i \rangle$ is an element of the context, then to satisfy this constraint the case must contain a pair $\langle a_i, V_i \rangle$ such that $V_i \in CV_i$.

In general, a null value for an attribute in a source case matches any value in an input case but is considered as a weaker match than a regular value. This is similar to the approach of intensional query interpretation (Lipski, 1979). We say that cases match definitely if no null values are present. The presence of null values changes it to a possible match. This extends the notions of equality and partial equality discussed earlier. If a null value is specifically included in a context, then there is no difference in matching a value or a null value.

Using various scenarios to specify constraints for a context is similar to, but more general than, a nearest-neighbor approach. Traditionally, nearest-neighbor algorithms can find cases close to a

given one, i.e., cases that would be indistinguishable if one would use a more coarse representation. One can imagine that the neighborhood of a case is a sphere around a point in the multidimensional space of attributes that are used to describe it. Our approach supports shaping such a neighborhood along certain dimensions.

There are several possible scenarios for creating the initial context and their advantages/disadvantages depend on the actual application. In the simplest approach, which is similar to query-by-example, a context Ω is constructed from an input case C_{input}. This process maps all attribute names from the C_{input} into attribute names in the Ω, and all attribute values from the C_{input} into the set of attribute-value constraints in the Ω:

$$\langle a_i : V_i \rangle \in C_{input} \leftrightarrow \langle a_i : \{V_i\} \rangle \in \Omega.$$

The initial context may be changed subsequently (either automatically by the system or by the user) as a reaction to a returned answer.[2] This approach can also be considered a starting point for other, more sophisticated, approaches.

If the CBR system cooperates with an underlying system (diagnostic system, planner, etc.) then available diagnosis and the state of a device may specify the context for a given task (referred to as task-based retrieval). In other words, the system may infer important attributes (salient features) as a subset of all the available attributes. Moreover, attribute values may be constrained.

A machine-learning or knowledge-mining algorithm could also be used to select important attributes for a given task and to specify characteristic values for them. This information can then be used to derive an initial context and specify context modification strategies.

If the CBR system is used as an intelligent decision support system, then the user is an expert who can improve the system's performance (in terms of both system competence and efficiency) by selecting important features and posing constraints on attribute values. Since the context can be changed dynamically, the user may start the request using all the available attributes and later remove certain irrelevant features (this approach is referred to as retrieval-by-reformulation). In summary, an initial context may be specified by the user, defined from an input case, or with created with help of a machine learning algorithm. Next we show how a given context can be used to retrieve relevant cases.

Traditional CBR systems locate similar cases on the basis of the problem at hand. In Figure 3.2 we show a context that was generated on the basis of a problem description. Attribute-value pairs are unchanged from the problem description except for the following:

- The *AGE* attribute extends matches to a neighborhood around the original value.

[2] In general, an answer consists of the set of relevant cases retrieved.

- The *DIAGNOSIS2ND* attribute is ignored.
- For the *NO_CYCLE* attribute any value larger than the given value is allowed.
- The *ABORT* attribute is allowed any value (it would be possible to specify that any of the three possible values in the domain = { *Yes, No, Not-Known*} counts for a match).

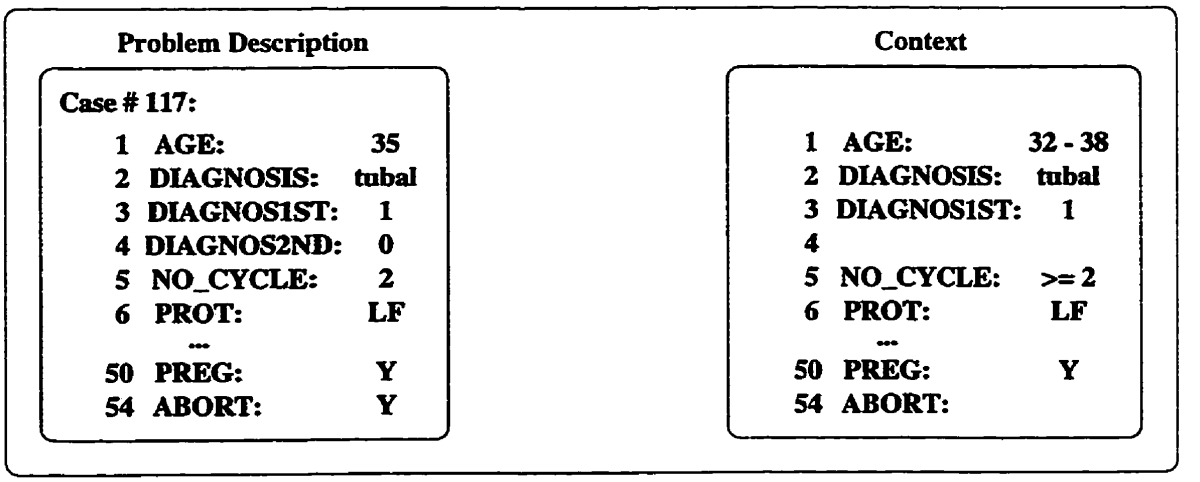


Figure 3.2: *Defining context on the basis of a problem case.*

We say that a case satisfies (or matches) a particular context if for each attribute specified in the context the value of that attribute in the case satisfies the constraint (i.e., the value is contained in the constraint set). The matching occurs on the attribute-value level, but the matching process can be affected by the way how individual attributes are combined into categories. Namely, context relaxation and iterative retrieval are affected by categories. This will be further described in Section 3.3.

Definition 3.2.4 (Satisfiability) *A case* $\mathcal{C}$ **satisfies** *a context* Ω, *denoted* $sat(\mathcal{C}, \Omega)$, *if and only if for all pairs* $\langle a_i : CV_i \rangle \in \Omega$, *there exists a pair* $\langle a_i : V_i \rangle \in \mathcal{C}$ *such that* V_i *is in* CV_i:

$$sat(\mathcal{C}, \Omega) \textit{ iff } \forall a_i (\langle a_i : CV_i \rangle \in \Omega \rightarrow \exists V_i \langle a_i : V_i \rangle \in \mathcal{C} \wedge V_i \in CV_i).$$

We say that the context Ω *is* **satisfiable** *if and only if there exists at least one case in a case base that satisfies it:*

$$\Omega \texttt{ is satisfiable } \textit{iff } \exists \mathcal{C} \in \Delta \ : \ sat(\mathcal{C}, \Omega).$$

Later we will show that this definition can be relaxed using an *m-of-n* matching (Ortega, 1995). This would make a case satisfy a context if it matches m of its attributes, where $m < n$, rather than requiring all of the n attribute constraints in the context be satisfied.

The representation of a specific problem and a solution depends on the available case representation scheme and on the task the case is going to be used for. Context defines which aspects of a case are important in a particular situation. Thus, individual contexts define views of source cases. Figure 3.3 depicts a satisfiable context and shows a given case view in that context. Figure 3.4 shows an un-satisfiable context. Not surprisingly, in different situations, one view might be suited more to support a problem-solving process than alternatives. Our theory supports a flexible view changing (see Sections 3.3 and 5.4).

Definition 3.2.5 (Case view) *A* **view** *of a case C in context Ω, denoted C_Ω, is the subset of a case C which satisfies the context:*

$$\forall a_i(\langle a_i : V_i\rangle \in C_\Omega \textit{ iff } \langle a_i : V_i\rangle \in C \ \wedge \ \langle a_i : CV_i\rangle \in \Omega \ \wedge \ V_i \in CV_i).$$

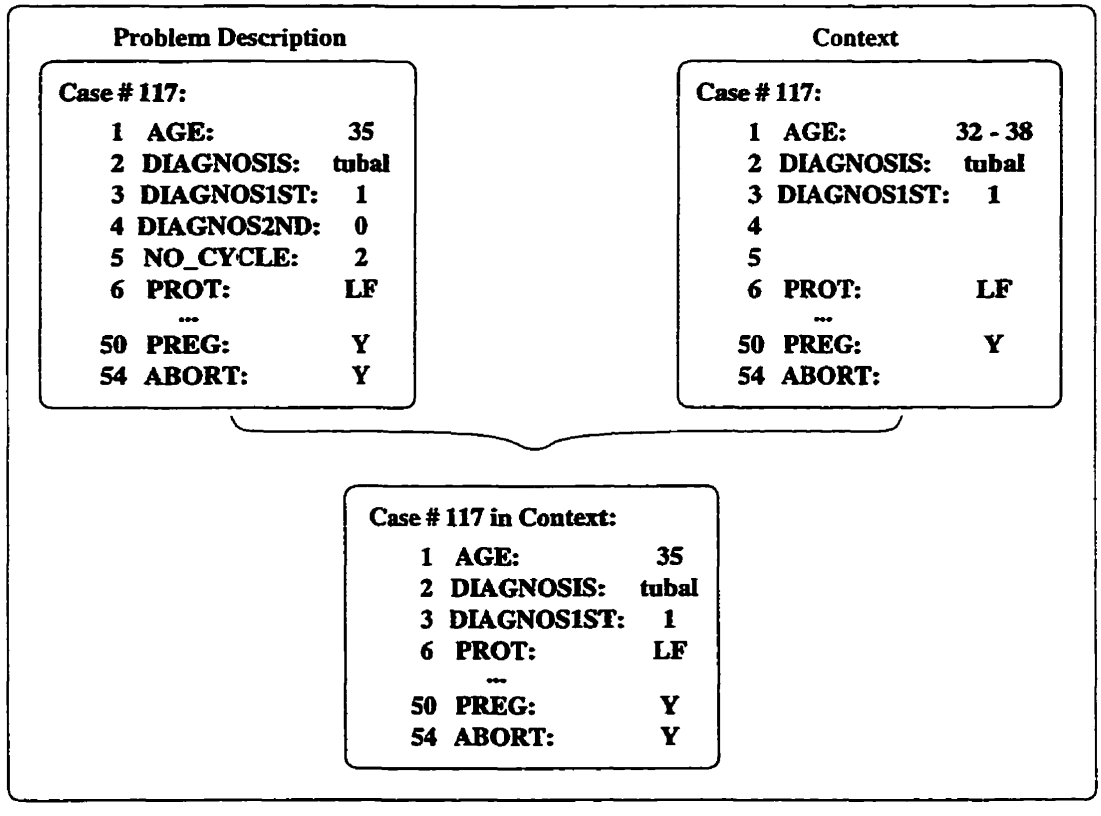


Figure 3.3: *Viewing a case in a satisfiable context.*

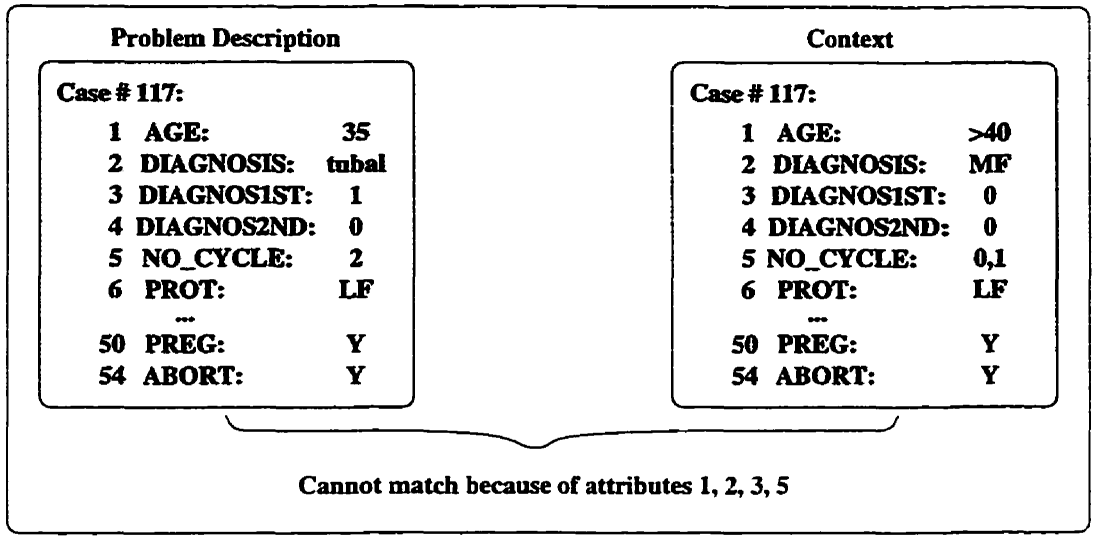


Figure 3.4: *An un-satisfiable context that cannot be used to view a given case.*

3.3 Context Transformations

Closeness of retrieved cases is controlled by explicitly defined context. If too many or too few relevant cases are retrieved using the initial context, then the system automatically transforms the context or lets the user modify it manually. There are two possible transformations: *relaxation* – to retrieve more cases and *restriction* – to retrieve fewer cases. These context transformations are a foundation for supporting iterative retrieval and browsing. On the basis of the theory introduced. here we discuss specific examples of relaxation and restriction transformations.

The relaxation technique can be used to return an answer to a specific query as well as related answers (Gaasterland, 1993). Without such automatic relaxation, users would need to submit alternative queries. The restriction technique works analogously.Since the search for relaxed or restricted contexts can be infinite, there must be a mechanism to control it, either by the user or by other means. We discuss these issues in Chapter 5.

The more transformations needed on one or more features in a query, the less similar the retrieved cases are to the original query (Suzuki, Ohnishi and Shigemasu, 1992). There are several specific transformations that can lead to the relaxation and/or the restriction of a context. First we define relaxation and restriction of context and then we discuss their possible implementations.

Definition 3.3.1 (Relaxation) *A context Ω_1 is a* **relaxation** *of a context Ω_2, denoted $\Omega_1 \succ \Omega_2$. if and only if the set of attributes for Ω_1 is a subset of the set of attributes for Ω_2 and for all attributes in Ω_1, the set of constraints in Ω_2 is a subset of the constraints in Ω_1. As well, contexts Ω_1 and Ω_2 are not equal.*

$$\Omega_1 \succ \Omega_2 \textit{ iff } \forall \langle a_i : CV_i \rangle \in \Omega_1,\ \exists \langle a_i : CV_j \rangle \in \Omega_2\ :\ CV_i \supseteq CV_j\ \wedge\ \Omega_1 \neq \Omega_2.$$

Thus, relaxation is performed by either reducing the number of attributes required during matching, or by extending the constraint set of a given context (i.e., increasing the interval or number of possible values, or decreasing the number of prohibited values).

Definition 3.3.2 (Restriction) *A context Ω_1 is a* **restriction** *of a context Ω_2, denoted $\Omega_1 \prec \Omega_2$, if and only if Ω_2 is a relaxation of Ω_1:*

$$\Omega_1 \prec \Omega_2 \textit{ iff } \Omega_2 \succ \Omega_1.$$

In other words, restriction is performed by either increasing the number of attributes required during matching, or by reducing the constraint set of a given context.

An obvious application for context restriction and relaxation is during iterative case retrieval. We need to define how context transformations affect satisfiability, i.e., how the set of cases that satisfy context, changes with relaxation and restriction respectively.

Theorem 3.3.1 *If Ω_1 is a relaxation of Ω_2 then all cases that satisfy Ω_2 also satisfy Ω_1:*

$$\forall \mathcal{C} \in \Delta : (sat(\mathcal{C}, \Omega_2) \wedge \Omega_1 \succ \Omega_2) \rightarrow sat(\mathcal{C}, \Omega_1).$$

Proof:

Assume for a given case $\mathcal{C}$, $sat(\mathcal{C}, \Omega_2)$ and $\Omega_1 \succ \Omega_2$.

From Definition 3.2.4 and $sat(\mathcal{C}, \Omega_2)$:

$$\forall a_j \langle a_j : CV_j \rangle \in \Omega_2 \rightarrow \exists V_j \langle a_j : V_j \rangle \in \mathcal{C} \wedge V_j \in CV_j. \quad (1)$$

From Definition 3.3.1 and $\Omega_1 \succ \Omega_2$:

$$\forall \langle a_i : CV_i \rangle \in \Omega_1, \exists \langle a_i : CV_j \rangle \in \Omega_2 : CV_i \supseteq CV_j \wedge \Omega_1 \neq \Omega_2. \quad (2)$$

Assume for some a_i: $\langle a_i : CV_j \rangle \in \Omega_1$.

From (1): $\exists V_j \langle a_i : V_j \rangle \in \mathcal{C} : V_j \in CV_j$. (3)

From (2) and (3): $V_j \in CV_i$. (4)

From (3) and (4):

$$\exists V_j \langle a_i : V_j \rangle \in \mathcal{C} \wedge V_j \in CV_j. \quad (5)$$

Therefore:

$$\forall a_i \langle a_i : CV_j \rangle \in \Omega_1 \rightarrow \exists V_j \langle a_i : V_j \rangle \in \mathcal{C} \wedge V_j \in CV_j. \quad (6)$$

Thus by Definition 3.2.4: $sat(\mathcal{C}, \Omega_1)$.

□

Corollary 3.3.1 *If Ω_1 is a restriction of Ω_2 then the set of cases that satisfy Ω_1 is a subset of the set of cases that satisfy Ω_2:*

$$\forall \mathcal{C} \in \Delta : (sat(\mathcal{C}, \Omega_1) \wedge \Omega_1 \prec \Omega_2 \rightarrow sat(\mathcal{C}, \Omega_2)).$$

Proof:

Trivial from Definition 3.3.2 and Theorem 3.3.1.

□

3.3.1 Transformations for Context Relaxation

Definition 3.3.1 implies two transformations for context relaxation, *reduction* and *generalization*. The former removes constraints by reducing the number of attributes required to match from m to p, where $0 < p < m \leq n$. Generalization relaxes the context by enlarging the set of allowable values for an attribute. One of the following approaches can be used: adding values, extending intervals, or generalizing values to classes using a generalization hierarchy.

Definition 3.3.3 (Reduction) *Context Ω_1 is a* **reduction** *of context Ω_2 if and only if Ω_1 is a subset of attribute-value pairs for Ω_2.*

$$\texttt{reduction}(\Omega_1, \Omega_2) \textit{ iff } \Omega_1 \subset \Omega_2.$$

Because attributes can be grouped into categories, each category can be reduced separately. Thus, some categories may require all attributes to match, while attributes in other categories may be ignored completely, or attributes in some categories may use *m-of-n* matching. In Section 5.3 we define a function `reduce (Ω, Category)`, which returns a context Ω' by modifying specified category in Ω, i.e., reducing the number of attributes required to match.

Definition 3.3.4 (Generalization) *Context Ω_1 is a* **generalization** *of Ω_2 if and only if for all attributes in Ω_1 and Ω_2, the set of constraints for attributes in Ω_2 is a subset of the constraints for attributes in Ω_1:*

$$\texttt{generalization}(\Omega_1, \Omega_2) \textit{ iff } (\forall a_i (\exists CV_i \langle a_i : CV_i \rangle \in \Omega_1) \leftrightarrow (\exists CV_j \langle a_i : CV_j \rangle \in \Omega_2))$$

$$\wedge\ (\forall a_i, CV_i, CV_j (\langle a_i : CV_i \rangle \in \Omega_1 \ \wedge\ \langle a_i : CV_j \rangle \in \Omega_2) \rightarrow CV_j \subset CV_i).$$

Generalization is performed on attribute level. If no specific attribute is defined for relaxation, then the first attribute in the category with a lowest priority is used. In Section 5.3 we define a function **generalize** (Ω, **Attribute**), which returns a context Ω' by generalizing the specified attribute in Ω.

Theorem 3.3.2 *Reduction is a form of relaxation:*

$$\texttt{reduction}(\Omega_1, \Omega_2) \rightarrow \Omega_1 \succ \Omega_2.$$

Proof:

Assume $\langle a_i : CV_i \rangle \in \Omega_1$.

From Definition 3.3.3 and definition of $\subset$: $\langle a_i : CV_i \rangle \in \Omega_2$.

Thus, $\exists \langle a_i : CV_j \rangle \in \Omega_2 \ \wedge \ CV_j \subseteq CV_i$.

From Definition 3.3.3: $\Omega_1 \neq \Omega_2$.

Therefore: $\forall \langle a_i : CV_i \rangle \in \Omega_1, \ \exists \ \langle a_i : CV_j \rangle \in \Omega_2 : CV_i \supseteq CV_j \wedge \Omega_1 \neq \Omega_2$.

From Definition 3.3.1 it follows that $\Omega_1 \succ \Omega_2$.

□

It should be noted that despite the possible range of values for p, only p "close to" m makes sense. If p is significantly smaller than m then a loose matching criterion is defined. It was shown that setting $p = m - 2$ helps to achieve the highest ratio of correct to incorrect answers (Ortega, 1995). In Chapter 5 we present techniques that can be used and controlled to enhance the power of the retrieval component.

Assume the following attributes in a category of a case:

source_code	C++
project_size	20K
programmer	Martin

If no reduction is in effect, all attributes have to match for the category to match. Reduction specifies that only p of m attributes in a category need to match a case to satisfy a context.

Theorem 3.3.3 *Generalization is a form of relaxation:*

$$\texttt{generalization}(\Omega_1, \Omega_2) \rightarrow \Omega_1 \succ \Omega_2.$$

Proof:

Assume $\langle a_i : CV_i \rangle \in \Omega_1$.

From Definition 3.3.4: $\langle a_i : CV_j \rangle \in \Omega_2 \ \wedge \ CV_i \supset CV_j$.

Thus, $\Omega_1 \neq \Omega_2$.

Therefore: $\forall \langle a_i : CV_i \rangle \in \Omega_1, \ \exists \ \langle a_i : CV_j \rangle \in \Omega_2 : CV_i \supseteq CV_j \wedge \Omega_1 \neq \Omega_2$.

From Definition 3.3.1 it follows that $\Omega_1 \succ \Omega_2$.

□

In order to explain how context can be relaxed using generalization, assume the following attribute constraints:

source_code	{C, C++}
project_size	{20K}

Generalization specifies more allowed values, such as adding Pascal for the source_code attribute:

source_code	{C, C++, Pascal}
project_size	{20K}

3.3.2 Transformations for Context Restriction

Definition 3.3.2 implies two transformations for context restriction, *expansion* and *specialization*. The former strengthens constraints by enlarging the number of attributes required to match: given *m-of-n* matching, the required number of attributes is increased from m to p, where $0 \leq m < p \leq n$. Thus, expansion is the inverse transformation to reduction. Specialization strengthens constraints by removing values from a constraint set for an attribute. This may lead to a decreased number of cases that satisfy the resulting context.

Definition 3.3.5 (Specialization) *Context* Ω_1 *is a* **specialization** *of* Ω_2 *if and only if* Ω_1 *is a generalization of* Ω_1*:*

$$\texttt{specialization}(\Omega_1, \Omega_2) \ \textit{iff} \ \texttt{generalization}(\Omega_2, \Omega_1).$$

Specialization is performed on attribute level. If no specific attribute is defined for restriction, then the first attribute in the category with a lowest priority is used. In Section 5.3 we define a function `specialize (Ω, Attribute)`, which returns a context Ω' by specializing the specified attribute in Ω.

Definition 3.3.6 (Expansion) *Context* Ω_1 *is an* **expansion** *of* Ω_2 *if and only if* Ω_2 *is a reduction of* Ω_1*:*

$$\texttt{expansion}(\Omega_1, \Omega_2) \ \textit{iff} \ \texttt{reduction}(\Omega_2, \Omega_1).$$

Because attributes can be grouped into categories, each category can be restricted separately. Thus, some categories may require all attributes to match, while attributes in other categories may be ignored completely, or attributes in some categories may use *m-of-n* matching. In Section 5.3 we define a function **expand** (Ω, **Category**), which returns a context Ω' by modifying specified category in Ω, i.e., expanding the number of attributes required to match.

Theorem 3.3.4 *Expansion is a form of restriction:*

$$\texttt{expansion}(\Omega_1, \Omega_2) \rightarrow \Omega_1 \prec \Omega_2.$$

Proof:

Trivial from Definition 3.3.6, Theorem 3.3.2 and Definition 3.3.2.

In order to explain context restriction through expansion, assume the following attributes in a category of a case:

source_code	C++
project_size	20K
programmer	Martin

Initially, the context is satisfied if only one out of the three attributes matches. This constraint can be strengthened by requiring that either two out of the three attributes match, or all three attributes match. Strengthening the constraint lowers the number of cases that satisfy the context. Thus, context satisfiability depends on m in *m-of-n* matching. To illustrate this, consider the following context, which is satisfiable with $m = 2$ but not with $m = 3$:

source_code	{Pascal}
project_size	{20K}
programmer	{Martin}

Theorem 3.3.5 *Specialization is a form of restriction:*

$$\texttt{specialization}(\Omega_1, \Omega_2) \rightarrow \Omega_1 \prec \Omega_2.$$

Proof:

Trivial from Definition 3.3.5, Theorem 3.3.2 and Definition 3.3.2.

□

Specialization strengthens the context by removing values from a constraint set for an attribute. Thus, it reduces the number of cases that can satisfy the context. Consider the following case and context:

Case		*Context*	
source_code	C++	source_code	Language
project_size	20K	project_size	{20K}
programmer	Martin	programmer	{Martin}

Specializing the source_code attribute constraint from Language to {C, Pascal} eliminates the case from the set of relevant cases if $m = 3$ is used. The case would only qualify for matching if $m < 3$.

3.4 Context Operations

In order for a formalism to deal with context changes, we define the notions of context addition and difference.[3] These operations define set-theoretic union and difference of constraints on attribute values of contexts. This supports interpreting context transformations and expressing incremental context modifications as described in Section 5.4.

Definition 3.4.1 (Context addition) *Given contexts* Ω_1 *and* Ω_2, $\Omega_1 + \Omega_2$ *is a new context that is defined as the set-theoretic union:*

$$\langle a_i : CV_k \rangle \in \Omega_1 + \Omega_2 \text{ iff } \langle a_i : CV_i \rangle \in \Omega_1 \wedge \langle a_i : CV_j \rangle \in \Omega_2 \wedge CV_k = CV_i \cup CV_j.$$

Definition 3.4.2 (Context difference) *Assuming contexts* Ω_1 *and* Ω_2, $\Omega_1 - \Omega_2$ *is a new context that is defined as the set-theoretic difference:*

$$\langle a_i : CV_k \rangle \in \Omega_1 - \Omega_2 \text{ iff } \langle a_i : CV_i \rangle \in \Omega_1 \wedge \langle a_i : CV_j \rangle \in \Omega_2 \wedge CV_k = CV_i - CV_j.$$

Using the above definitions, we can express context transformations in terms of context operations. *Reduction* removes an attribute a_k with constraints on its value CV_k either *permanently* – an attribute a_k is removed from the context: $\Omega' = \Omega - \{\langle a_k : CV_k \rangle\}$ – or *dynamically* – an *m-of-n* matching is used: $\Omega' = \Omega_0 + \Omega_1 + \ldots + \Omega_l$.[4] Ω_i represents combinations of attribute value pairs.

[3] Similarly, we can define a context product, to generate a context that specifies (or groups) more than one cluster of objects (Mylopoulos and Motschnig-Pitrik, 1995).

[4] l is computed as $l = \binom{n}{n-k}$.

Thus, for Ω', the set of retrieved cases $SI' = SI_0 \cup SI_1 \cup \cdots \cup SI_l$, where SI_i are sets of returned cases for combinations of attribute value pairs Ω_i. *Expansion* adds an attribute a_k with constraints on its value CV_k: $\Omega' = \Omega + \{\langle a_k : CV_k \rangle\}$.

Generalization weakens the constraint for a given attribute by enlarging the set of allowable values: $\Omega' = \Omega + \{\langle a_k : V \rangle\}$, where V can be single value or a set of values. *Specialization* restricts a constraint by removing values from a constraint set for an attribute: $\Omega' = \Omega - \{\langle a_k : V \rangle\}$, where V can be single value or a set of values.

3.5 Similarity Relation

In the theory proposed, similarity is considered a relation between two cases.[5] It is defined as a supplement to equivalence to enable partial matches. In order to control the level of the partial match, we define similarity with respect to a given context. Variable-context similarity assessment is an integral part of retrieval in the proposed case-based reasoning system.

Definition 3.5.1 (Similarity) *A case C_1 is* **similar** *to a case C_2 with respect to a given context Ω, denoted $C_1 \sim_\Omega C_2$, if and only if both C_1 and C_2 satisfy context Ω:*

$$C_1 \sim_\Omega C_2 \leftrightarrow sat(C_1, \Omega) \wedge sat(C_2, \Omega).$$

The similarity relation can be extended to a relation on a set of cases (Jantke, 1994). Assume a set of cases, SI, a context Ω, and a case base Δ. The relation $relevant(SI, \Omega, \Delta)$ states that the cases contained in set $SI \subseteq \Delta$ are relevant with respect to a context Ω. We say that this relation holds when SI is defined as the set of cases that satisfy the given context:

$$relevant(SI, \Omega, \Delta) \text{ iff } \forall C \in \Delta, C \in SI \leftrightarrow sat(C, \Omega).$$

From the definition of similarity, all cases that satisfy the context are similar to each other and all are relevant to the input problem. Figure 3.5 exemplifies case relevance with respect to the context. Thus, $Case_1$, $Case_3$ and $Case_4$ are relevant to the context. This principle is used during case retrieval: Specifying a context and a case base, the task is to retrieve all relevant cases, i.e, all cases that are similar with respect to a given context.

[5] Similarity is a relation not a function – it does not require that every element in the domain be mapped into exactly one element in the range.

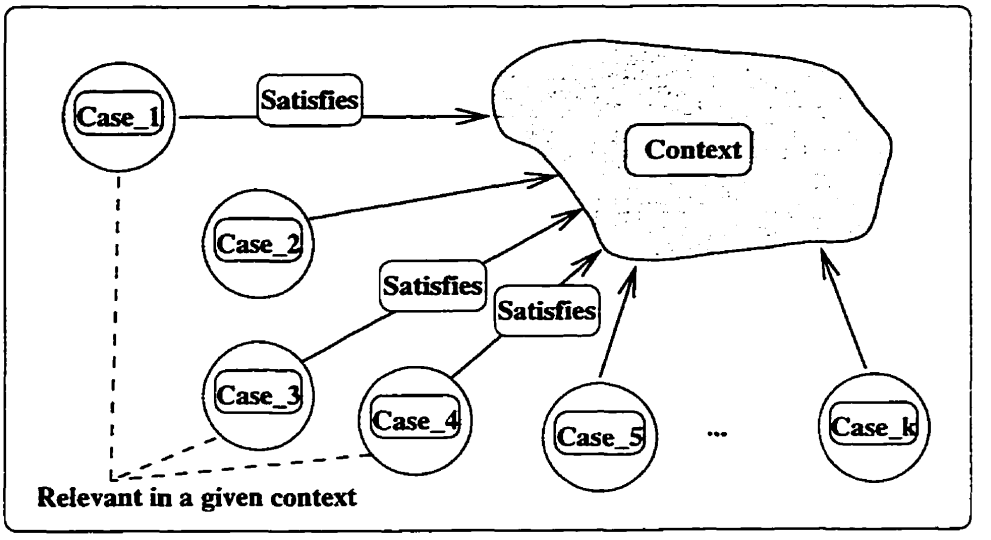


Figure 3.5: *Relevant set of cases.*

3.5.1 Temporal Aspects of the Similarity Relation

Because knowledge is dynamic, similarity assessment must be dynamic as well. Over a period of time the representation we may see changes in the content of a case base or the context. Time need not be an integral part of the similarity definition, since it is naturally captured in the context and in the representation of cases.

If a case representation, a context or a case base is changed between time τ_i and τ_j for cases $\mathcal{C}_1$ and $\mathcal{C}_2$ then the similarity of $\mathcal{C}_1$ and $\mathcal{C}_2$ may change. If a case representation, a context and a case base are unchanged between time τ_i and τ_j for cases $\mathcal{C}_1$ and $\mathcal{C}_2$, then the similarity of $\mathcal{C}_1$ and $\mathcal{C}_2$ is unchanged as well:

$$(\mathcal{C}_1 \sim_\Omega \mathcal{C}_2)_{\tau_i} \equiv (\mathcal{C}_1 \sim_\Omega \mathcal{C}_2)_{\tau_j}$$

Because the problem, the case base, and the user's preferences may change over time, similarity of cases cannot be predefined. On the basis of these changes, different cases would satisfy the current context, different attributes would need to match, etc. Thus, the following are prone to changes as well: (1) relevance of cases in a case base, (2) relevance of attributes that represent cases, and relevance of attributes used during the retrieval, (3) relevance of values, and (4) relevance of retrieved cases to a given problem.

Motivated by examples such as those presented in Section 3.1.2, we define three kinds of dependency of similarity assessment on context: context-dependent, permanently and temporally context-independent similarity.

Definition 3.5.2 (Permanently context-independent similarity) *Similarity assessment becomes permanently context-independent if regardless of changes in the explicit context, similarity depends*

only on the implicitly assumed context.[6]

If a CBR system uses indexing to find similar cases and the indices are context-independent, then it can assess similarity regardless of any changes. Thus, the system measures only context-independent similarity. This approach works if the context for measuring similarity does not change over time, which happens when the task remains the same.

Definition 3.5.3 (Temporally context-independent similarity) *Similarity assessment becomes temporally context-independent if similarity is assessed for the same objects during a short time interval τ. Then, there is a psychologically explainable tendency to use the old context in a new situation (Barsalou, 1989). The length of τ depends on the strength of such dependency, frequency of the repetition and other factors. After enough occurrences of temporal independence we could have permanently context-independent similarity.*

Systems that support temporally context-independent similarity are usually based on the notion of recency, i.e., more recent cases are valued more than older cases, even if they share less attributes with the problem case.

Definition 3.5.4 (Context-dependent similarity) *Similarity assessment is context-dependent if it is modified with changed (explicitly stated) context.*

Systems that support context-dependent similarity subsume both previous approaches. On the one hand, if the same context is specified all the time, the system models the permanently context-independent similarity. On the other hand, more recent cases can still maintain an advantage over more similar but older cases. Moreover, explicitly defined context supports (1) iterative browsing with the ability to explain the relevance of retrieved items, and (2) reasoning process in multi-context environments. Thus, context improves retrieval flexibility and increases relevance of cases retrieved by the system.

3.5.2 Retrieve Function

The *retrieve* function is a basis for recalling relevant cases – given a context Ω and a case base Δ the function returns the set of cases $SI \subseteq \Delta$ that are relevant to the given context.

Definition 3.5.5 (Retrieve Function) *Given a case base Δ and a context Ω, the retrieve function returns a set of relevant cases $SI \subseteq \Delta$ such that all cases in SI satisfy the given context. (If the context Ω is not satisfiable then the retrieve function returns an empty set.)*

$$\texttt{retrieve}(\Omega, \Delta) = SI \textit{ iff } \textit{relevant}(SI, \Omega, \Delta).$$

[6] An explicit context refers to the context defined in Section 3.2.2. An implicit context refers to the context that is assumed by the user.

The basic retrieval algorithm that implements the retrieve function is presented in Figure 3.6. The complexity of this naive retrieval function is characterized by the number of comparisons required, i.e., the total number of cases multiplied by the number of comparisons required for a given context:[7] $\mathcal{O}(\mid \Omega \mid \times \mid \Delta \mid)$.

```
retrieveNaive (Context, CaseBase)
    initialize Answer to ∅
    for all cases in the CaseBase
        retrieve (Case_i)
        if  Case_i satisfies the Context then
            add (Case_i, Answer)
    end
    return (Answer)
end
```

Figure 3.6: *Naive retrieval algorithm. Context is initialized with the attributes and constraints from the input case.*

This algorithm can be extended in two ways. First, by implementing an iterative context relaxation and restriction. Second, by using an incremental context transformation. The former extension is presented in Figure 3.7. The latter extension is described in Section 5.4.

The algorithm presented in Figure 3.7 uses two bounds on the number of returned cases - lower and upper limit. The former is used as a condition to apply context relaxation. The latter is used as a condition to apply context restriction. Functions `relax` (Ω, Category) and `restrict` (Ω, Category) return a relaxed or restricted initial context. (In Chapter 5 we describe how the user can affect automatic context transformations.)

3.5.3 Explain Function

Given a case base Δ and a set of cases SI, the *explain* function returns a context Ω that specifies constraints on attributes such that all cases in SI satisfy them. The explain function is a basis for explanation-based learning (Greiner and Jurisica, 1992; Minton et al., 1989; Segre and Elkan, 1994), a technique that creates a generalized explanation for some given knowledge. The explain function can be used to find useful features (the context) in order to explain a given set of cases. This problem is also referred as feature selection (Cardie, 1996).

There is a sizeable number of possible contexts that explain a given set of cases. This is caused by the existence of a large number of attribute-value pairs that could be used to cluster cases. A solution to this problem is to limit the number of possible contexts the algorithm considers. Since the user is interested in finding out only strong patterns in the case base, an obvious approach is

[7] In Section 5.4 we show how efficiency of the retrieval algorithm can be improved by using an incremental context transformation algorithm adapted from database management systems.

```
iterativeRetrieveNaive (Context, CaseBase, LowerLimit, UpperLimit)
    initialize Answer to ∅
    add (retrieveNaive (Context, CaseBase), Answer)
    if  | Answer | ≤ LowerLimit then
        set Context' to relax (Context, Category)
        iterativeRetrieveNaive (Context', CaseBase, LowerLimit, UpperLimit)
    else if | Answer |> UpperLimit then
        if the Context was not previously relaxed then
           set Context' to restrict (Context, Category)
           iterativeRetrieveNaive (Context', CaseBase, LowerLimit, UpperLimit)
    else return (Answer)
end
```

Figure 3.7: *Naive iterative retrieval algorithm. Context is initialized with the attributes and constraints from the input case. Special counters are used to prevent repeating context restrictions and relaxations forever.*

to maximize the number of attributes and minimize the number of values in the selected contexts. In other words, the explain function searches for the most restricted context the cases satisfy. The most restricted context is characterized by inclusion of the highest number of possible attributes with the minimal number of different values defined for individual attributes.

To make the task of searching for the context manageable and more user-centered we extend the explanation function with two variations. The first variation is based on the idea that trying to find a context that is satisfied by all cases may be unreasonable and/or not needed. Thus, a user can specify a simpler task instead, i.e., find a context that is satisfied by $X\%$ of all cases, where X is the user-specified threshold for context satisfiability. Second, a user poses a query that includes only a subset of all possible cases. Thus, instead of considering all patients, only those with certain characteristics are used as input to the explain function. The explain function then explores only a relevant subset of all possible attributes. Since a user explores possible contexts by specifying different sets of cases and various thresholds, the explain function can be viewed as a visualization tool. It was observed that visualization-based knowledge mining is a more successful approach than automatic knowledge mining (Kohavi, 1997). We now formally define the explain function.

Definition 3.5.6 (Explain Function) *Given a case base Δ and a set of cases $SI \subseteq \Delta$, the explain function returns the most restrictive context Ω such that all (and only) cases in SI satisfy the returned context.*

$$\mathbf{ex}(SI, \Delta, \Omega) \textit{ iff } \forall C \in SI \rightarrow sat(C, \Omega)$$

$$\mathbf{explain}(SI, \Delta) = \Omega \textit{ iff}$$

$$\mathbf{ex}(SI, \Delta, \Omega) \ \wedge (\forall \Omega_i (\mathbf{ex}(SI, \Delta, \Omega_i) \rightarrow \Omega \preceq \Omega_i)).$$

```
explain (SetOfCases, CaseBase)
  initialize the Context to include all attribute-values
        for all cases included in the SetOfCases
    return (Context)
end
```

Figure 3.8: *Explain function algorithm. As defined above, the SetOfCases is s subset of the CaseBase.*

Theorem 3.5.1 *The explain function algorithm presented in Figure 3.8 implements the explain function described in Definition 3.5.6.*

Proof:

Assume that all attribute-values for all cases $\mathcal{C} \in SI$ are used to generate the context Ω.

From Definition 3.2.4: $\forall \mathcal{C} \in SI,\ sat(\mathcal{C}, \Omega)$. (1)

From (1) and Definition 3.5.6: $\mathbf{ex}(SI, \Delta, \Omega)$. (2)

From (2) and Definition 3.3.1: $\forall \Omega_i(\mathbf{ex}(SI, \Delta, \Omega) \rightarrow \Omega \succeq \Omega_i$. (3)

From (2) and (3): $\mathbf{explain}(SI, \Delta) = \Omega$. (4)

Therefore from (4), the algorithm from Figure 3.8 implements the explain function of Definition 3.5.6. □

The complexity of the explain function (see Figure 3.8) depends on the number of cases considered and the number of features used to describe them: $\mathcal{O}(\mid \mathcal{C} \mid \times \mid SI \mid)$, where $\mid \mathcal{C} \mid$ represents the maximum number of attributes of a case and $\mid SI \mid$ represents the number of cases in set SI.

Often it is better to have a more restricted context than to achieve 100% satisfiability. In Figure 3.9 we present a modified explain function algorithm, that explores the space of possible contexts. The algorithm finds a context that uses the maximum number of attributes with the minimum number of constraints on them. The main idea is to use a user-defined threshold UT, as a stopping criterion for satisfiability of a context. In other words, instead of requiring that all cases in SI satisfy the context, it is sufficient if $UT\%$ of cases in SI satisfy the context.

3.6 Features of Similarity

This section presents features of the proposed similarity assessment theory. These features are essential when applying the retrieval and explanation similarity functions. Even though similarity is

```
explain (SetOfCases, CaseBase, UserThreshold)
  initialize the Context to include all attribute-values
      for all cases included in the SetOfCases
  initialize SatisfiabilityValue of a given context to 100%
  for all attributes in the Context
      while SatisfiabilityValue > UserThreshold
          specialize (Context)
          set SatisfiabilityValue based on how many cases in SetOfCases
              satisfy the restricted Context
      end
  end
  return (Context)
end
```

Figure 3.9: *A modified explain function algorithm. SatisfiabilityValue of a given context measures how many cases in a SetOfCases satisfy a given context. Context specialization is aimed at removing the least frequent attribute-values for a given attribute in the context.* $|\,.\,|$ *means set cardinality.*

neither symmetric nor transitive in general, context helps to determine when similarity is transitive, symmetric, monotonic and distributive. These properties can advantageously be used during relevance assessment.[8]

3.6.1 Distributivity

In order to design an efficient incremental algorithm that deals with context transformations (which is introduced in Section 5.4), we need the retrieve function to satisfy distributivity. This feature is needed to guarantee that incremental changes can be combined in order to provide a correct answer for the complete query. The following theorem proves distributivity of the retrieve function.

Theorem 3.6.1 (Distributivity) *Assuming arbitrary contexts* Ω_i *and* Ω_j*, the retrieve function is distributive:*

$$\texttt{retrieve}(\Omega_i + \Omega_j, \Delta) \;=\; \texttt{retrieve}(\Omega_i, \Delta) \cap \texttt{retrieve}(\Omega_j, \Delta).$$

Proof:

Assume for some $C \in \Delta$, $C \in \texttt{retrieve}(\Omega_i + \Omega_j, \Delta)$.

From Definition 3.5.5: $sat(C, \Omega_i + \Omega_j)$. (1)

From Definition 3.4.1:

$\langle a_m : CV_m \rangle \in (\Omega_i + \Omega_j)$ *iff*

$\langle a_m : CV_i \rangle \in \Omega_i \;\wedge\; \langle a_m : CV_j \rangle \in \Omega_j \;\wedge CV_m = CV_i \cup CV_j.$ (2)

[8] The same properties can be stated for dissimilarity. Although dissimilarity (as a measure of irrelevance) is not symmetric in general (Subramanian and Genesereth, 1987), our approach helps to determine the circumstances where dissimilarity is symmetric, using the same theory and reasoning as in Section 3.6.3. Similarly as in Section 3.6.2, it can be shown that variable-context dissimilarity is monotonic. However, it can be shown that dissimilarity is not reflexive and transitive.

From (1) and (2), and Definition 3.2.4:

$sat(C, \Omega_i) \wedge sat(C, \Omega_j)$. (3)

From (3) and Definition 3.5.5:

$C \in retrieve(\Omega_i, \Delta) \wedge C \in retrieve(\Omega_j, \Delta)$. (4)

Using set theory and (4):

$C \in (retrieve(\Omega_i, \Delta) \cap retrieve(\Omega_j, \Delta))$. (5)

The converse is proved by reversing the above argument, i.e., from (5) to (1).

Thus, $C \in \texttt{retrieve}(\Omega, \Delta)$ *iff* $C \in \texttt{retrieve}(\Omega_i, \Delta) \cap \texttt{retrieve}(\Omega_j, \Delta)$.

Therefore from (5): $\texttt{retrieve}(\Omega, \Delta) = \texttt{retrieve}(\Omega_i, \Delta) \cap \texttt{retrieve}(\Omega_j, \Delta)$.

□

3.6.2 Monotonicity

Monotonicity is an important property for flexible case retrieval – as more constraints are put in the query, fewer (or at most the same number of) cases satisfy it. We will show that the retrieve function defined using variable-context similarity assessment is monotonic with an inverse monotonicity – more constraints imply fewer retrieved cases and vice versa.

Theorem 3.6.2 (Monotonicity) *The retrieve function based on variable-context similarity assessment is monotonic. More precisely, given a case base* Δ *and contexts* Ω_1, Ω_2*:*

$$(\Omega_1 \succ \Omega_2) \rightarrow retrieve(\Omega_1, \Delta) \supseteq retrieve(\Omega_2, \Delta).$$

$$(\Omega_1 \prec \Omega_2) \rightarrow retrieve(\Omega_1, \Delta) \subseteq retrieve(\Omega_2, \Delta).$$

Proof:

Assume $\Omega_1 \succ \Omega_2$.

For any given C, if $C \in retrieve(\Omega_2, \Delta)$, then $sat(C, \Omega_2)$ by Definition 3.5.5. (1)

From Theorem 3.3.1, $\Omega_1 \succ \Omega_2$ and (1): $sat(C, \Omega_1)$. (2)

From Definition 3.5.5 and (2): $C \in retrieve(\Omega_1, \Delta)$. (3)

Therefore from (3): $(\Omega_1 \succ \Omega_2) \rightarrow retrieve(\Omega_1, \Delta) \supseteq retrieve(\Omega_2, \Delta)$.

Directly from above: $\Omega_1 \succ \Omega_2 \rightarrow \Omega_2 \prec \Omega_1$.

Therefore $(\Omega_1 \prec \Omega_2) \rightarrow retrieve(\Omega_1, \Delta) \subseteq retrieve(\Omega_2, \Delta)$.

□

Monotonicity feature is useful during the retrieval process (Rissland et al., 1993; Lauzon and Rose, 1994; Jurisica, 1994). On the one hand, if not enough cases have been retrieved from the case

base, the initial context may be relaxed to retrieve additional "less" similar cases. On the other hand, if too many cases are retrieved, the initial context may be restricted to lower the number of retrieved cases. Restricting the context increases the similarity among retrieved cases, since more attributes are required to match.

3.6.3 Symmetry

Several researchers (Tversky, 1977; Smith, 1989; Gentner and Bowdle, 1994; Ohnishi, Suzuki and Shigemasu, 1994; Jantke, 1994; Ricci and Avesani, 1995) have argued that not all similarities are symmetric. Our claim is that relevance judgments (based on context and similarity) are symmetric if the context is preserved.[9] An unchanged context is, however, an important part of our claim. This requirement is needed because of Definitions 3.2.4, 3.3.1 and 3.3.2. Partial *m-of-n* matches are allowed only if m is not changed and if we treat the group of attributes on which the *m-of-n* match applies as a group of attributes and not as individual ones.

Consider a comparison of North Korea to China (Tversky, 1977; Gentner and Bowdle, 1994). On the basis of psychological experiments Tversky argues that North Korea is more similar to China than China to North Korea. Based on what attributes and constraints are contained in the context (political system and/or geography), it is easy to see that stating the context explicitly makes similarity symmetric: The political system of China is similar to that of North Korea and vice versa. It is only when the context is assumed implicitly or we use additional information (e.g., which country had a particular political system in place earlier) that similarity becomes asymmetric.

As another example, compare children to parents. On the basis of psychological experiments (Smith and Heise, 1990), parents are not compared to children and thus the similarity is not symmetric. However, stating the context explicitly, for example the eye color, makes the comparison of children to parents equivalent to the comparison of parents to children. This leads to making a distinction between perceptual and conceptual similarity (Smith and Heise, 1990). Perceptual similarity is dynamic and changes based on our beliefs about objects. Smith and Heise consider conceptual similarity to be knowledge that is explicit. Thus, explicitly defined context changes dynamic perceptual similarity (which is asymmetric) to conceptual similarity (which is symmetric).

Theorem 3.6.3 (Symmetry) *Given the context and similarity definitions, relevance judgments are symmetric if the context is not changed:*

$$C_1 \sim_\Omega C_2 \leftrightarrow C_2 \sim_\Omega C_1.$$

[9] Carnap and Ruspini share similar views (Carnap, 1967; Ruspini, 1991).

Proof:

From Definition 3.2.4 $C_1 \sim_\Omega C_2 \leftrightarrow sat(C_1, \Omega) \wedge sat(C_2, \Omega)$. (1)

From (1): $\leftrightarrow sat(C_2, \Omega) \wedge sat(C_1, \Omega)$ (2)

From (2): $\leftrightarrow C_2 \sim_\Omega C_1$. (3)

Therefore $C_1 \sim_\Omega C_2 = C_2 \sim_\Omega C_1$.

□

A context can be relaxed or restricted using the Definitions 3.3.1 and 3.3.2. Although the resulting set of cases that satisfy the context may be enlarged or reduced by respective context transformations, the symmetry relation is preserved if all cases are considered in a current context.

3.6.4 Transitivity

Transitivity seems to be the most controversial feature of similarity. Goodman, Tversky and Smith are only a few of the many who argue that similarity is not transitive (Goodman, 1951; Tversky, 1977; Smith, 1989). However, we have already shown that context plays a central role in similarity judgments and thus affects transitivity (see Section 3.1.2). Here we show that relevance (based on context and similarity) is transitive if the context is preserved. We show how an altered context may result in non-transitivity of relevance assessment.

All examples presented in (Tversky, 1977) have an implicit context which is changed if the subject and referent are exchanged; thus, making these examples non-transitive on the first look.

Example 3.6.1 *Jamaica is similar to Cuba (because of geographical proximity); Cuba is similar to Russia (because of their political affinity); but Jamaica and Russia are not similar at all. Suppose one wants to retrieve all countries similar to Cuba. If a geographical context is specified, Jamaica would be one of the retrieved countries; if political affinity is the context, Russia would be one of the retrieved countries. It is apparent that applying different contexts to the same information base may result in retrieving different cases.*

Theorem 3.6.4 (Transitivity) *Variable-context similarity assessment is transitive:*

$$C_1 \sim_\Omega C_2 \wedge C_2 \sim_\Omega C_3 \rightarrow C_1 \sim_\Omega C_3.$$

Proof:

Assume $C_1 \sim_\Omega C_2$ and $C_2 \sim_\Omega C_3$.

From Definition 3.5.1: $sat(C_1, \Omega) \wedge sat(C_2, \Omega) \wedge sat(C_3, \Omega)$. (1)

From (1) and Definition 3.2.4 it follows that: $C_1 \sim_\Omega C_3$.

Therefore $C_1 \sim_\Omega C_2 \wedge C_2 \sim_\Omega C_3 \rightarrow C_1 \sim_\Omega C_3$.

□

A context can be relaxed or restricted using the Definitions 3.3.1 and 3.3.2. Although the resulting set of cases that satisfy the context may be enlarged or reduced by respective context transformations, the transitivity relation is preserved if all cases are always considered in a current context. All cases that satisfy a given context are relevant in that context, as illustrated in Figure 3.5.

3.6.5 Reflexivity

Reflexivity of similarity is a trivial relation: since a case is identical to itself, it is also similar to itself. Hence, the similarity is a reflexive relation. Variable-context similarity assessment is also reflexive. Recall that the context must be satisfiable; otherwise, there is no point of discussing reflexivity of variable-context similarity assessment.

Theorem 3.6.5 (Reflexivity) *Variable-context similarity assessment is reflexive with respect to a satisfiable context* Ω: $\mathcal{C} \sim_{\Omega} \mathcal{C}$.

Proof:

Assume: $sat(\mathcal{C}, \Omega)$.

From Definition 3.5.1: $\mathcal{C} \sim_{\Omega} \mathcal{C}$ *iff* $sat(\mathcal{C}, \Omega)$. (1)

From (1), Definition 3.2.4, and the assumption
that the context is satisfiable it follows that $\mathcal{C} \sim_{\Omega} \mathcal{C}$.

Therefore: $\mathcal{C} \sim_{\Omega} \mathcal{C}$.

□

3.7 Discussion

There are usually three types of similarity defined: semantic similarity, structural similarity and pragmatic similarity. They differ mainly on feature selection for similarity assessment.

Semantic similarity is defined as a correspondence to the meaning of compared cases. Structural similarity involves comparing the representation structure between cases, i.e., the relationships between individual attributes are compared.

Pragmatic similarity uses only a particular set of features during matching. This set of features defines what is relevant and what is irrelevant. Our notion of context fits this description by explicitly biasing the retrieval toward selecting relevant cases. Case relevance is determined using a satisfiability definition.

Generally, CBR systems do not represent context explicitly. Instead, the context is encoded into the case base by defining attribute weights that are used during the similarity computation. There are several problems with this approach. First, if the relevance assessment is based on the equivalence of cases in a predefined context, the system is single-context oriented. Thus, the system

cannot provide a correct or satisfactory answer for diverse users or in different situations. Second, if the relevance assessment is expressed only numerically then the system cannot semantically specify the relevance of the returned answer to the current task. This makes explanation and automatic query relaxation and restriction more difficult. Third, if the relevance assessment is based on the partial matches of cases, it becomes non-transitive. Fourth, predefined relevance links (indices) make the system single-task. Thus, schemas from various case bases cannot be easily integrated, in order to build cooperative CBR systems. Predefined relevance links also prohibit analogical mappings.

These problems motivated us to use a non-standard approach to retrieval and similarity assessment in case-based reasoning. This chapter proposed a novel approach to similarity (and relevance) assessment that helps to solve the problems mentioned above. Relevant cases are retrieved using variable-context similarity assessment, which supports: (1) flexibility by automatic query restriction or relaxation, (2) schema mappings, analogical reasoning and knowledge interchange by defining explicit context, and (3) multi-user and multi-task reasoning by considering user preferences and task specifications when defining a context. These features can be used to build multi-functional information bases. Moreover, the proposed theory helps to determine the circumstances where the similarity is distributive, reflexive, symmetric, transitive and monotonic.[10]

A wide range of applications can take an advantage of variable-context similarity assessment and its features (see Section 6 for respective examples). In addition, the proposed simple, yet powerful, approach to matching can be implemented efficiently, as will be shown in Chapter 7.

Our definition of similarity relates to definitions of relevance and irrelevance (Subramanian and Genesereth, 1987; Greiner, 1994; Shimony, 1993; Levy, 1993; Levy and Sagiv, 1993; Levy, 1994). Retrieved cases are considered relevant if they are similar to the query with respect to a given context. All cases that are relevant to a given request can be treated as equivalent. The remaining cases are considered irrelevant with respect to a given context. Variable-context relevance assessment is a relation, since it does not require that every element in a domain is mapped into exactly one element in the range. In other words, it does not require that every case from a case base be similar to only one case with respect to a given context.

Context is the main part in the definitions of similarity, relevance, equality and equivalence. Cases C_1 and C_2 are *similar* or *relevant* if and only if $C_1 \sim_\Omega C_2$. Cases C_1, C_2 are *equivalent* if and only if they are either similar or equal. Cases C_1 and C_2 are *logically equivalent* if and only if for all contexts, if Ω is an interpretation of both C_1 and C_2, $C_{1\Omega} = C_{2\Omega}$. $\simeq$ is an equivalence relation and $C_1 \simeq C_2$ *iff* $C_1 \vdash C_2 \wedge C_2 \vdash C_1$. *Equivalence classes* group indistinguishable elements. Thus, cases that satisfy the same context form an equivalence class for a given context. It should be noted that for different contexts, different equivalence classes may be formed and cases may belong to multiple equivalence classes.

[10] It is possible to state the same features for irrelevance, but this is beyond the scope of this thesis.

Chapter 4

Using Variable-Context Similarity

The purpose of this chapter is to introduce a system, TA3, which uses the theory presented in Chapter 3 to describe an application of the variable-context similarity assessment to the medical domain. We show how context affects retrieval of relevant cases. For this purpose we have selected an in vitro fertilization domain, where the task is to suggest hormonal therapy for a patient on the basis of experience from previous patients, and to predict the outcome of IVF treatment. Performance evaluation of the system is presented in Section 6.2.1.

4.1 Introduction to In Vitro Fertilization

Almost one out of ten couples in North America is infertile. One of the most widely used treatments for human infertility is the *in vitro* fertilization (IVF) procedure. IVF is a fairly complex assisted reproductive technology, which has been evolving since the first pregnancy achieved by this method in 1978. Although certain parts of the procedure have been improved over the years, the pregnancy rates have not changed and are at best 30%.

The IVF procedure consists of patient selection by diagnosis of infertility, controlled ovarian stimulation for multiple oocyte recruitment and maturation, close monitoring of follicular development by ultrasound and hormonal assessment, oocyte retrieval, insemination of oocytes *in vitro*, determination of fertilization, assessment of embryo development and quality, assessment of endometrial quality, and intrauterine transfer of one or more cleaved embryos. At each step of the procedure, there are many variables, both dependent and independent, which may impact the chance of a successful outcome, i.e., pregnancy (see (Davis and Rosenwaks, 1995) for a review of *in vitro* fertilization). For example, a patient's response to controlled ovarian stimulation may depend on her age, diagnosis of infertility, size of ovaries, baseline serum follicle stimulating hormone concentration, type and dose of fertility drug used and whether endogenous gonadotrophin secretion has been

suppressed by gonadotrophin releasing hormone (GnRH) agonist. Pregnancy rates may depend on age, number of oocytes retrieved, presence of oocyte dysmorphisms, sperm quality, fertilization rate, percent cleavage, rate of cleavage and embryo quality, number of embryos transferred, endometrial thickness and uterine blood flow on ultrasound and number of previous cycles of treatment. With so many variables, it is difficult for the clinician to discern trends and make informed decisions to optimize success rates for each individual infertile couple.

There are many factors that influence the outcome of IVF. Intelligent decision support systems may enable IVF practitioners to cope with the complexity of the domain during treatment planning and help them discover relationships between individual knowledge sources, which can then be used to potentially improve the pregnancy rate. Since the IVF treatment is a relatively expensive procedure, such a decision system helps the patient in deciding whether to go ahead with the procedure or not, by finding similar past patients. The system finds both successful and unsuccessful past patients, share sufficient amount of descriptors.

The two most common tasks performed by physicians are as follows:

1. **Treatment suggestion and outcome prediction** – The physician performs this task in two stages:

   - Having initial information about the patient (first 8 attributes), the first stage involves finding similar patients from the information base and suggesting how to treat the current patient to increase the probability of successful pregnancy. This includes finding all similar cases, and using retrieved cases with pregnancy as successful examples and retrieved cases without pregnancy as negative cases. An adaptation process uses this information to predict attribute-values for the current case, including pregnancy outcome.
   - After the initial treatment is completed, additional attributes are available (first 39 attributes). The second stage involves predicting the outcome of the whole treatment, i.e., predicting the values for the remaining attributes, including pregnancy outcome. It should be noted that since more attribute-values are used during similar case retrieval, the pregnancy outcome could be changed, because different set of cases may be similar to the current patient.

2. **Patient data analysis** – Knowledge-mining techniques are used to find regularities in the case base. The physician has no particular case in mind, however, (s)he may consider the whole case base or only a certain case. The knowledge mining in $\mathcal{TA}3$ involves finding a context, in which particular group of cases is considered similar. The user can specify a threshold, which controls the quality (degree of relevance) and quantity of discovered information.

4.2 Similarity-Based Retrieval for IVF

Decision making in IVF is usually based on a combination of the patient's particular characteristics, and on the physician's knowledge and clinical experience. For most clinicians, the synthesis of previous experience with the current situation becomes almost intuitive with time. This phenomenon is commonly referred to as "clinical judgment." However, as the number of variables increase, the ease and accuracy of this approach decreases. A reasoning system based on past experiences, a CBR system, may be used as a computational aid in this decision making.

Our objective is to apply and evaluate the variable-context similarity-based retrieval algorithm in the IVF domain[1] The main goals are as follows:

1. Create a case base of patients who have been assessed for infertility and treated by means of IVF procedure using the $\mathcal{TA}3_{IVF}$ system.
2. Use the case base for suggesting an optimal (or close to optimal) approach for hormonal stimulation for new patients and predict pregnancy outcome (**suggestion/prediction**).
3. Use the case base to derive interesting relationships among data (**knowledge mining**).

4.3 Case Representation in IVF

Case acquisition is carried out by storing IVF patient medical records in the case base (see Figure 4.1 for an IVF case example). The case base, considered for this paper, consists of 788 cases with 55 attributes per case.[2] Out of 788 cases, there are 149 clinically successful pregnancies, 10 are pregnancies with ovarian hyper-stimulation syndrome (OHSS) complication, 12 are pregnancies ended by abortion, 4 are ectopic pregnancies (implantation occurred outside of uterus) and 632 are unsuccessful pregnancies. There is no information about IVF outcome in 7 cases. Based on the number of patients treated in the past, the expected growth of the case base is 1000 cases per year. In addition, as our experience shows, the number and selection of attributes used during problem solving is bound to increase as a result of accommodating new knowledge about factors influencing pregnancy outcome (Jurisica and Shapiro, 1995; Jurisica et al., 1998).

Cases in $\mathcal{TA}3_{IVF}$ are represented as frames. Attributes are grouped into Telos-style categories (Mylopoulos et al., 1990), as is necessary for performing individual tasks. Using the domain knowledge and the most frequently used contexts (e.g., for prediction in stage one, prediction in stage two and for evaluation) the system classifies cases into clusters to allow for efficient access.

[1] We refer to this system as $\mathcal{TA}3_{IVF}$, an implemented system that is described in Chapter 5. Performance evaluation of the system is presented in Section 6.2.1.

[2] For some experiments we have used a more complex case representation, namely a series of estrogen levels was instead of a single value. This changed the count for number of attributes to 70.

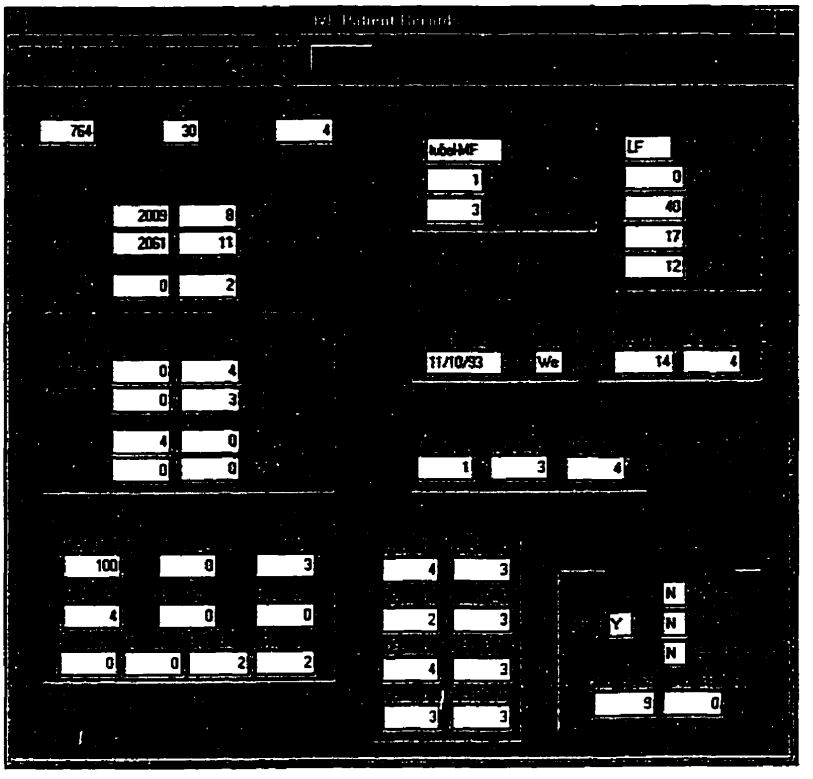


Figure 4.1: *Case base browser. IVF case is a set of attribute-value pairs, organized into categories.*

There are two types of classification – user-based and system-based. The former type is used when there is heuristic knowledge available. Such classification covers, for example, clustering patients into three age categories or clustering patients according to pregnancy outcome. System-based classification extracts information from the knowledge base automatically. Here, the system finds features and corresponding values that are good predictors for pregnancy and thus can be used as discrimination factors. We will show then how evaluation function can be used for such a task.

This approach can be viewed as a context-based classification.[3] The main advantage of this approach is its flexibility. In addition, the system supports explanation facility, because the explanation and relevance measure are based on the current context, not on the static and possibly old indexes. This does not mean that systems with implicitly represented context (via indexes) cannot work properly or that they cannot explain the reasoning process. It just means that $\mathcal{TA}3$ supports easier task switching, the needs of different users, and assesses similarity dynamically.

The domain vocabulary in $\mathcal{TA}3$ system uses a hierarchy of attribute values during the automatic context transformation (relaxation and restriction), and during the adaptation process. This does not mean that our techniques are not general. The information about the values is stored as an

[3] In CBR literature, this is similar to indexing. The main difference is that we use context explicitly, whereas most of the other systems assume context implicitly. Thus, context-based classification is more flexible, since it defines clusters dynamically.

external library of terms used in a particular domain. For a more efficient use, all terms are stored into hierarchies so the system can automatically select what can be used when and where.

4.4 The Adaptation Process for IVF

Adaptation process in CBR manipulates the solution of the source case to better fit the target case. This can be done automatically, semi-automatically or by the user. A domain knowledge or generally applicable adaptation rules can be used. The complexity of the domain and availability of domain knowledge dictates which approach can be used. It should be noted that adaptation is not possible in some domains due to lack of domain knowledge and complexity of the problem. Furthermore, there are domain where adaptation is not desirable at all because of liability reasons.

In $TA3_{IVF}$, a weighted average of attribute values from retrieved cases is computed automatically. Retrieved cases are ordered, first according to the class (pregnancy outcome), then according to a relevance measure, giving the most relevant cases the highest weight. If not enough cases are retrieved, automatic context relaxation is triggered. For example, if only two cases are retrieved and their values for *DAY_HCG* attribute are distant, the system relaxes current context to retrieve more cases. Then, a cluster of cases with an average value for a predicted attribute is formed to have the highest weight. Remaining retrieved cases have a marginal weight assigned. A weighted average is then computed for the predicted values. Although the system computes the suggested treatment, it is up to the physician to decide if the current patient does not have specific needs.

In the experiments conducted we used a requirement for the system to find five to fifteen cases and we allowed up to four context relaxations, if less than five cases are found. Attribute values (2-6, 50, 54) of case 117 were used as a context. The value for attribute 1 in context was set to be in the interval 27 to 40. Selected attributes have been grouped into two categories. Category 0 consisted of attributes 1, 50 and 54; the rest of the attributes comprised category 1. Initially, value constraint was set to *Some*(3) and *Some*(5) for categories 0 and 1 respectively.

The first returned answer is presented in Figure 4.2. Here, $recall_u = recall_q = 100\%$ and $precision_u = precision_q = 100\%$. Unless automatic relaxation is enabled by the user this is the final result. However, in the search for similar answers, context can be relaxed either using generalization or reduction. As a result, $precision_u$ would decrease while $recall_u$ would remain constant.

After applying reduction to category 1, additional cases are retrieved. Category 0 remained unchanged. Considering returned cases (see Figure 4.3) and the query, we have $recall_u = recall_q = 100\%$, $precision_q = 100\%$, and $precision_u = 25\%$.

Leaving category 0 unchanged and applying additional reduction to category 1 (see Figure 4.4) results in $recall_u = recall_q = 100\%$, $precision_q = 100\%$, and $precision_u = 16.7\%$.

In practice, however, the situation is more complex. There are usually two stages in a patient's

```
Case # 117:
 1          AGE:  35
 2    DIAGNOSIS:  tubal
 3   DIAGNOS1ST:  1
 4   DIAGNOS2ND:  0
 5     NO_CYCLE:  2
 6         PROT:  LF
50         PREG:  Y
54        ABORT:  Y
```

Figure 4.2: *Retrieval function. Retrieved cases for unmodified context.*

```
Case # 92:
 1          AGE:  39
 2    DIAGNOSIS:  tubal
 3   DIAGNOS1ST:  1
 4   DIAGNOS2ND:  0
*5     NO_CYCLE:  4
 6         PROT:  LF
50         PREG:  Y
54        ABORT:  Y
```

```
Case # 117:
 1          AGE:  35
 2    DIAGNOSIS:  tubal
 3   DIAGNOS1ST:  1
 4   DIAGNOS2ND:  0
 5     NO_CYCLE:  2
 6         PROT:  LF
50         PREG:  Y
54        ABORT:  Y
```

```
Case # 575:
 1          AGE:  28
 2    DIAGNOSIS:  tubal
 3   DIAGNOS1ST:  1
 4   DIAGNOS2ND:  0
*5     NO_CYCLE:  1
 6         PROT:  LF
50         PREG:  Y
54        ABORT:  Y
```

```
Case # 583:
 1          AGE:  32
 2    DIAGNOSIS:  tubal
 3   DIAGNOS1ST:  1
 4   DIAGNOS2ND:  0
*5     NO_CYCLE:  1
 6         PROT:  LF
50         PREG:  Y
54        ABORT:  Y
```

Figure 4.3: *Retrieval function. Retrieved case after the first iteration of context relaxation.*

```
Case # 92:
 1          AGE:  39
 2    DIAGNOSIS:  tubal
 3   DIAGNOS1ST:  1
 4   DIAGNOS2ND:  0
*5     NO_CYCLE:  4
 6         PROT:  LF
50         PREG:  Y
54        ABORT:  Y
```

```
Case # 104:
 1          AGE:  32
*2    DIAGNOSIS:  endo
*3   DIAGNOS1ST:  2
 4   DIAGNOS2ND:  0
 5     NO_CYCLE:  2
 6         PROT:  LF
50         PREG:  Y
54        ABORT:  Y
```

```
Case # 117:
 1          AGE:  35
 2    DIAGNOSIS:  tubal
 3   DIAGNOS1ST:  1
 4   DIAGNOS2ND:  0
 5     NO_CYCLE:  2
 6         PROT:  LF
50         PREG:  Y
54        ABORT:  Y
```

```
Case # 575:
 1          AGE:  28
 2    DIAGNOSIS:  tubal
 3   DIAGNOS1ST:  1
 4   DIAGNOS2ND:  0
*5     NO_CYCLE:  1
 6         PROT:  LF
50         PREG:  Y
54        ABORT:  Y
```

```
Case # 583:
 1          AGE:  32
 2    DIAGNOSIS:  tubal
 3   DIAGNOS1ST:  1
 4   DIAGNOS2ND:  0
*5     NO_CYCLE:  1
 6         PROT:  LF
50         PREG:  Y
54        ABORT:  Y
```

```
Case # 611:
 1          AGE:  38
*2    DIAGNOSIS:  endo
 3   DIAGNOS1ST:  *2
 4   DIAGNOS2ND:  0
 5     NO_CYCLE:  2
 6         PROT:  LF
50         PREG:  Y
54        ABORT:  Y
```

Figure 4.4: *Retrieval function. Retrieved cases after second relaxation of context.*

treatment. During the first stage, first eight attributes out of 55 are known. The task to the system is to predict what kind of treatment is required in order to have a successful pregnancy, e.g., pregnancy (attribute 50), no OHSS (attribute 53), no abortion (attribute 54) and no ectopic (attribute 55). The second stage begins when the patient has finished the treatment and embryos are ready to transfer. Again, the question is to predict the success rate from cases similar in first 39 attributes. Next we describe competence evaluation of retrieval function, i.e., we evaluate the quality of solutions.

4.5 Retrieving Cases in IVF

Generally, information retrieval systems are optimized to achieve either high recall and moderate precision or vice versa. Precision can be explained as a measure of avoidance of irrelevant cases, while recall is a measure of completeness of retrieval. The flexibility of $\mathcal{TA}3_{IVF}$ helps to achieve precision- and recall-oriented retrieval, and retrieval where both recall and precision are high.

Given input information for a new patient the goal of $\mathcal{TA}3_{IVF}$ is to retrieve only cases highly relevant to the input case (the query). The retrieval component of the system is based on a modified nearest-neighbor matching (Wettschereck and Dietterich, 1995). Its modification includes:

- grouping attributes into categories of different priorities so that different preferences and constraints can be used for individual categories during query relaxation;
- using an explicit context during similarity assessment;
- using efficient query relaxation algorithm based on incremental context modifications.

The relevance of cases to a given request can be measured using various approaches. Since the usefulness of individual attributes varies, we define context as the set of attributes relevant for a given retrieval (based on a given task and user preferences), such as AGE, $CYCLE$, $DIAGNOSIS$, etc. Thus, the context can be seen as a view or an interpretation of a case, where only a subset of the attributes are considered relevant.

By selecting only certain attributes for matching and imposing constraints on attribute values, a context allows for controlling what can and what cannot be considered as a partial match: All (and only) cases that satisfy the specified constraints for the context are considered similar and are relevant with respect to the context. This allows for a controlled retrieval process, as well as for easy context transformations, such as context restriction and relaxation.

Similarity is determined as a closeness of values for attributes defined in the context (note that closeness depends on attribute value distribution as well as on the task being solved). These attributes can be specified directly by the user or a query-by-example may be used. Here, the context is defined through query-by-example and is used for an initial prediction (see Figure 4.5).[4] Since only

[4] If the patient's initial information is already in the case base, a query is constructed by simple case selection.

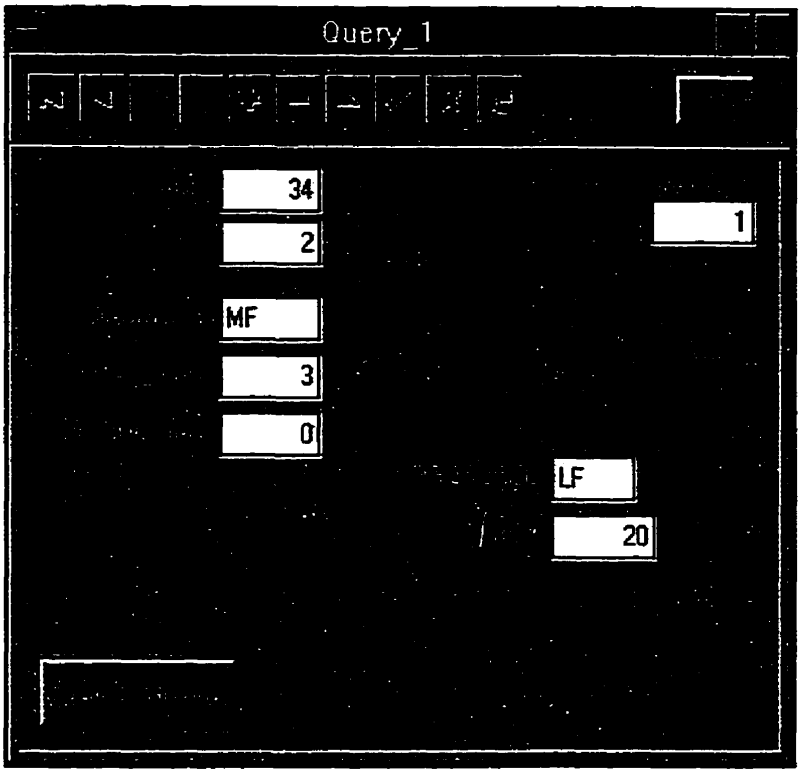


Figure 4.5: *Context for the first stage of prediction. Query-by-example editor. The query is formed either by browsing the case base and selecting a particular case or by entering a new case description. In both cases, $TA3_{IVF}$ returns cases similar to a given example.*

a subset of all attributes is required during matching (i.e., only a subset of all attributes is relevant for a given task), this context contains only attributes *AGE, CYCLE, DIAGNOSIS, FIRST_DIAG, SECOND_DIAG, PROTOCOL and BCP (birth control pill).* If the interpretation of the target case is similar to a source case, we say that they match and thus retrieved source cases are relevant in a given context. In addition, it is possible to impose constraints on the attribute values and on the number of attributes required to match (see Figure 4.6).

An explicitly defined context increases the flexibility of matching it supports. However, the problem is to specify the context. In the simplest approach, which is similar to query-by-example, a context Ω is constructed by mapping an input case into it in its entirety. The initial context may be subsequently changed (either by the system or by the user) as a reaction to a returned answer.[5] This approach can also be considered a starting point for other, more sophisticated, approaches. A machine-learning or knowledge-mining algorithm could also be used to select important attributes for a given task, and to specify characteristic values for them. This information can then be used to derive an initial context specification, and specify context transformation strategies. If the CBR system is used as a decision-support system, then the user is an expert who can improve the system's

[5] In general, an answer consists of a set of retrieved relevant cases.

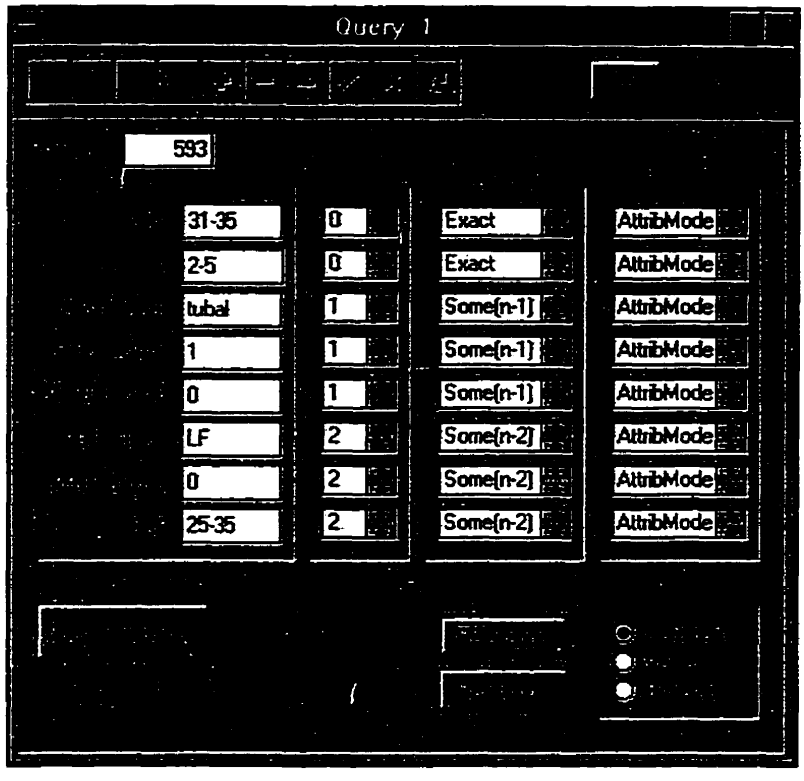


Figure 4.6: *Context for the first stage of prediction. In addition to specifying the case description, constraints on attributes can be defined.*

performance by selecting important features and posing specific constraints on attribute values.[6] Since the context can be changed dynamically, the user may start the request using all the available attributes and later remove certain irrelevant features (this approach is referred to as retrieval-by-reformulation). In addition, context may also contain important information about the problem, such as configuration of a specific device, input values from certain sensors, etc.

Since the number of tasks to be solved is limited and the number of users is bounded, it is feasible to use a caching mechanism for context, to support reuse of frequent contexts. This includes user preferences, attribute and attribute value constraint selections for individual tasks and relaxation criteria. Thus, without losing flexibility, $\mathcal{TA}3_{IVF}$ can be customized for a particular IVF clinic and for particular type of a user.

Monotonicity in context-based relevance (Jurisica and Glasgow, 1996a) enables us to control the number of retrieved cases by relaxing and restricting the context. If too many or too few relevant cases are retrieved using the initial context, then the system transforms the context or a user may modify it manually.

$\mathcal{TA}3_{IVF}$ supports two implementations of context relaxation – *reduction* and *generalization*. Recall that *reduction* removes constraints by reducing the number of attributes required to match:

[6] This improves not only competence of the system, but its efficiency as well.

given m_of_n matching, the required number of attributes is reduced from m to p, where $0 \leq p < m \leq n$. *Generalization* is a context transformation that relaxes the context by enlarging the set of allowable values for an attribute.

As an example, consider the relaxation of a context specified in Figure 4.7. In order to increase versatility, we grouped attributes into three categories. Assume that all categories have the same priority and value constraints but they require a different number of attributes to match. It is not possible to apply reduction on category 1, since there is only one attribute. Category 2 and 3 can be relaxed using reduction. *Some*(2), in the cardinality criteria for category 3, implies that at least two of the attribute-value pairs need to match for the whole category to match. This constraint could be further relaxed to *Some*(1). Additionally, category 2 could be relaxed from *Some*(2) to *Some*(1), as illustrated in Figure 4.7.

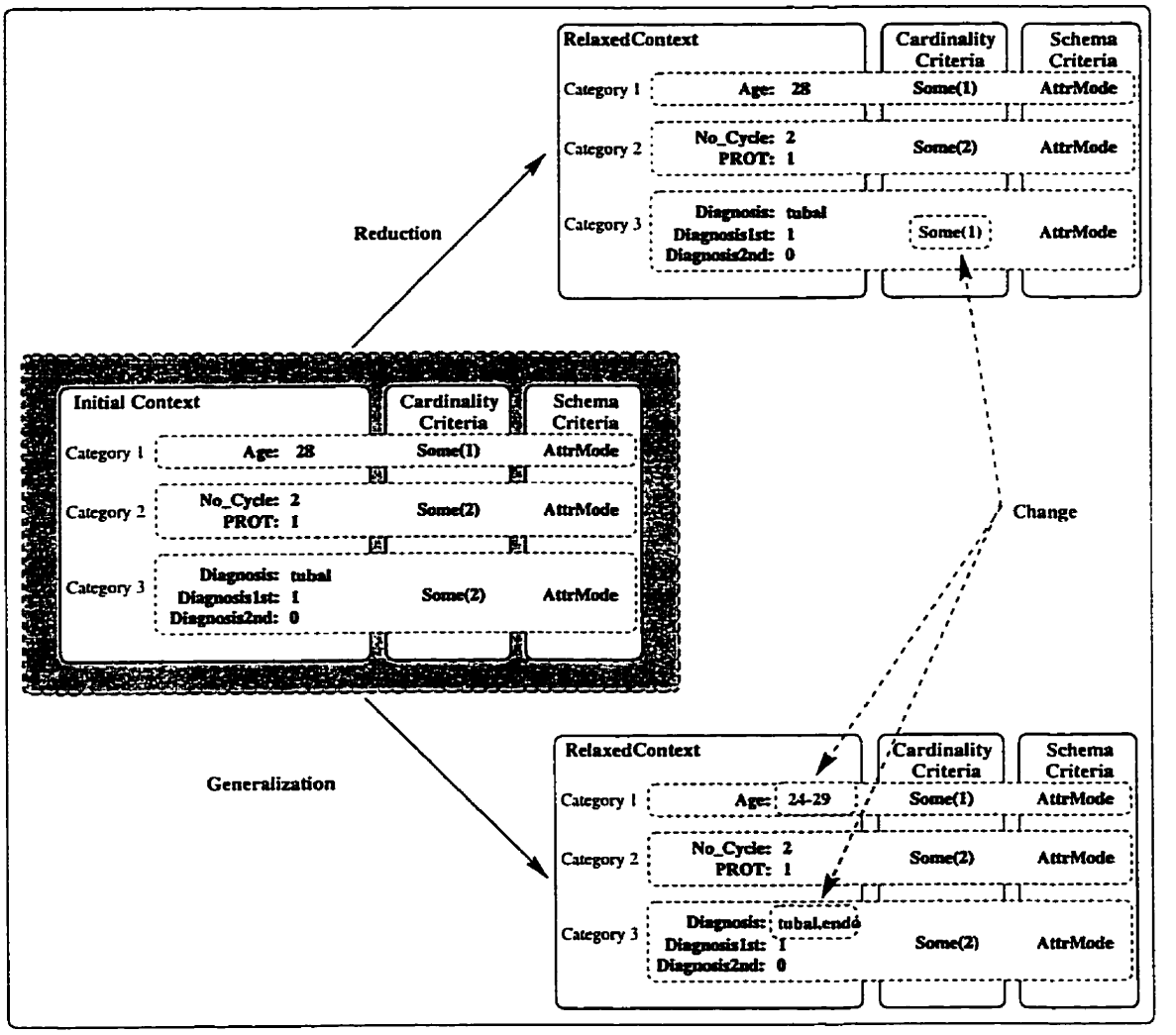


Figure 4.7: *Context relaxation by reducing the number of attributes required to match and by enlarging the set of allowable values for an attribute.*

In the second scenario, value constraints are relaxed. Since all categories required *instance*

matches, context values could be updated by using the conceptual hierarchy, presented in Figure 4.8. Thus, the constraint for the AGE attribute could be changed to $24-29$ (as shown in Figure 4.7), as an immediate generalization of 28. In addition, a user-guided relaxation may be a better option, since the user might have domain knowledge not represented in the system. Thus, the attribute value could alternatively be relaxed from 28 to for example, $26-30$. The constraint of attribute 1 in category 3 is also relaxed from constraint *tubal* to constraint set $\{tubal, endo\}$.

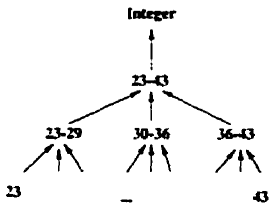


Figure 4.8: *Conceptual hierarchy for the AGE attribute.*

Contexts can also be iteratively restricted to retrieve successively fewer cases. There are two possible implementations of context restriction: *expansion* - strengthening constraints by enlarging the number of attributes required to match and *specialization* - strengthening constraints by removing values from a constraint set for an attribute (see Figure 4.9).

In the experiments conducted, we required the system to find five to fifteen cases and allowed up to four context relaxations (if less than five cases were found).

As mentioned earlier, the prediction task has two stages. During the first stage, the first eight (out of 55) attributes are known. The system's task is to predict the preferred treatment (attributes *NO_HMG, DAY_HCG*) in order to have a successful pregnancy, e.g., pregnancy, no OHSS, no abortion and no ectopic pregnancy. The second stage begins after a patient has finished the treatment and embryos are ready to transfer, when the task is to predict the success rate from cases similar in the first 39 attributes. To evaluate the retrieval process, we selected case 593 randomly as a target case and used the system to predict attribute values during the two stages. Figure 4.5 shows the context we selected in order to find relevant cases. Attributes *NO_HMG, DAY_HCG, PREG, OHSS, ABORT, ECTOPIC* illustrate the results of retrieval.

Table 4.1 lists all relevant cases after the first stage of prediction. Considering the retrieved cases, the adaptation algorithm predicts values for the remaining attributes, including the treatment outcome. Since the main goal is to have a pregnancy, the system favors the values suggested by cases 646, 662 and 666. Thus, the suggested amount of hormonal stimulation to be received (*NO_HMG*) is 18 (average value from cases 646, 662, 666). Similarly, the suggested day for triggering the ovulation (*DAY_HCG*) is 13 (average value from cases 646, 662, 666). If we look at the values of target case 593, the prediction error is small. Considering all cases, there is a predicted chance of 43% for pregnancy and 29% for pregnancy without complications. The next stage will give us a better comparison among cases selected during the first stage.

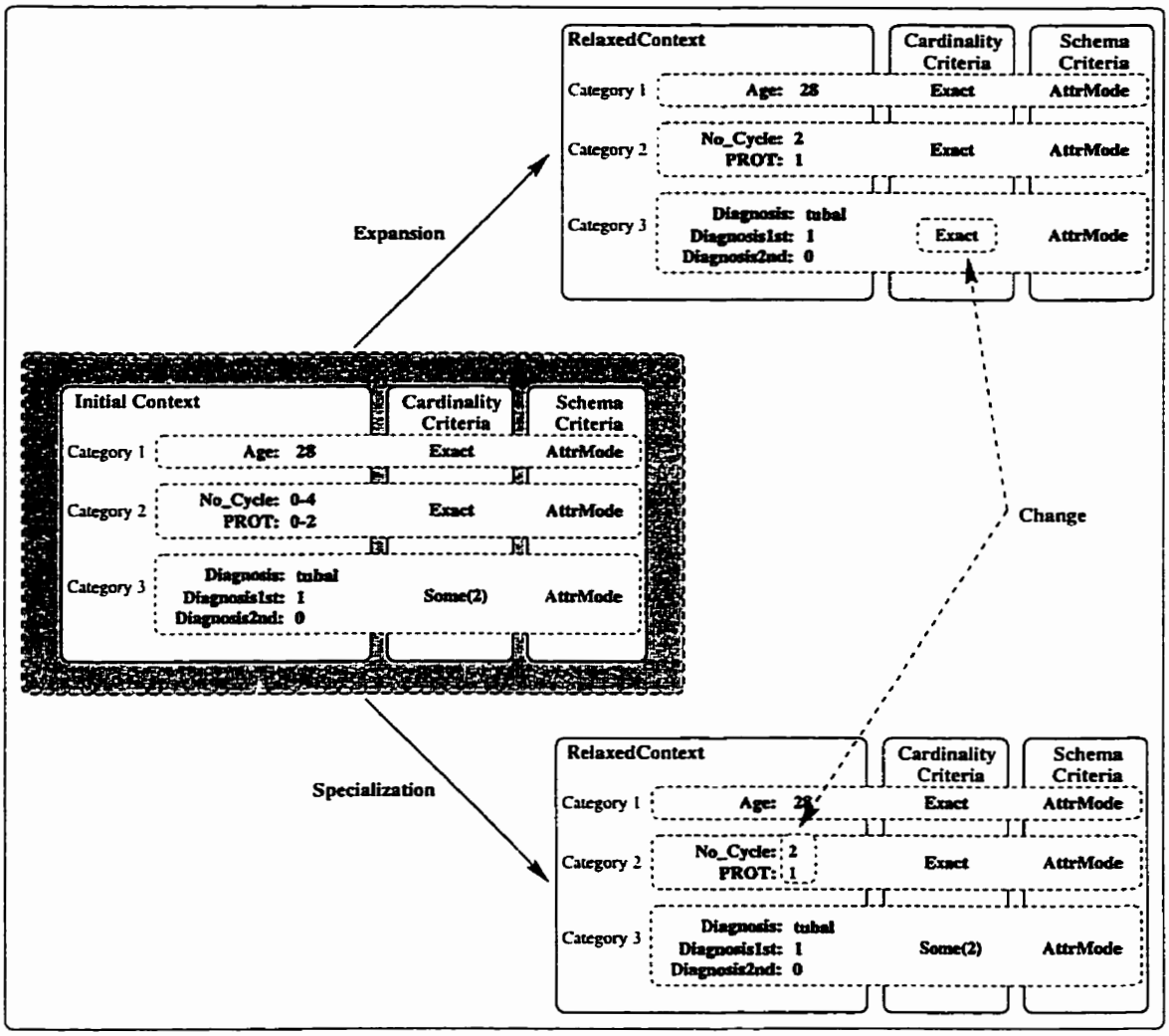


Figure 4.9: *Context restriction by enlarging the number of attributes required to match and by reducing the set of allowable values for an attribute.*

Attribute Name	*Similar Cases*							*SV*	*AV*
CASE #	533	555	573	638	646	662	666	-	593
AGE	32	34	33	31	32	31	32	-	32
DIAGNOSIS	tubal	tubal	tubal	tubal	tubal	tubal	tubal	-	tubal
DIAGNOS1ST	1	1	1	1	1	1	1	-	1
DIAGNOS2ND	0	0	0	0	0	0	0	-	0
NO-CYCLE	2	4	2	2	2	2	2	-	4
PROT	LF	LF	LF	LF	LF	LF	LF	-	LF
PROT1	0	0	0	0	0	0	0	-	0
BCP	27	27	35	36	37	25	31	-	27
NO-HMG	48	16	8	10	24	10	21	20	18
DAY-HCG	15	14	13	13	13	15	12	13	11
PREG	N	N	N	N	Y	Y	Y	Y	Y
OHSS	NA	NA	NA	NA	N	N	N	N	N
ABORT	NA	NA	NA	NA	N	Y	N	N	N
ECTOPIC	NA	NA	NA	NA	N	N	N	N	N

Table 4.1: *Accuracy of predicting hormonal therapy. We present suggested (SV) and actual (AV) values for the treatment, namely day of human chorionic gonadotrophin administration (DAY_HCG) and the number of ampoules of human menopausal gonadotrophin (NO_HMG).*

For the second stage of the prediction experiment, we extended a context to cover the first 39 attributes. After restricting the context, only 3 relevant cases were located (see Table 4.2). Using these cases, the system predicted a 100% chance of pregnancy and 33% chance of abortion.

Knowledge Mining in IVF

Knowledge mining in $\mathcal{TA}3_{IVF}$ is user-directed, i.e., user specifies the context and $recall_u$, which constrain the search space. The search space can also be constrained by specifying only a subset of the whole information base, as opposed to using the whole information base. Deploying a user-guided knowledge discovery process in $\mathcal{TA}3$ was motivated by: (1) the need to identify salient attributes for use in case-based classification; (2) the need to structure the case base into clusters of relevant cases for improving competence; and (3) the need to find representative values for salient attributes.

Pattern discovery in $\mathcal{TA}3_{IVF}$ involves collecting together cases that share something in common. For example, collecting all women that have pregnancy with complications and are in their third or fourth cycle. However, pattern identification alone is not sufficient - patterns also need to be described, i.e., given a set of cases, labeled by class (such as pregnant women), derive a description of the classes. Although the user can guide the search for patterns, from a statistical point of view, the process is an exploratory analysis, since no hypothesis about what the patterns may be is posed. Such a process of discovery is inherently dynamic and must often be performed iteratively.

$\mathcal{TA}3_{IVF}$ uses context reduction to help organize cases to support flexible retrieval, which may improve prediction accuracy and scalability. Moreover, new relationships among attributes may be

Attribute Name	*Similar Cases*			*Actual Case*
	666	662	646	593
AGE	32	31	32	32
DIAGNOSIS	tubal	tubal	tubal	tubal
DIAGNOS1ST	1	1	1	1
DIAGNOS2ND	0	0	0	0
NO-CYCLE	2	2	2	4
PROT	LF	LF	LF	LF
PROT1	0	0	0	0
BCP	31	25	37	27
NO-HMG	21	10	24	18
DAY-HCG	12	15	13	11
E2-HCG	7317	5071	11413	5208
ENDO-HCG	11	8	9	12
NOFOL	3	3	5	3
E2-FOL	0	0	0	0
E2–HCG1	12205	7153	11247	6373
ENDO-HCG1	11	8	9	11
CYC-OPU	14	17	15	13
TOT-EGGS	8	6	8	6
DON-EGG	0	0	0	0
EGG-INCU	8	6	8	6
IMM	2	0	2	3
IM	3	0	0	0
MAT	0	3	5	3
PM	3	3	1	0
CNA	0	0	0	0
SPERNO	3	7	5	3
SPMOT	0.9	0.9	0.9	0.9
SPQUAL	4	4	3	3
EGG-FER	4	5	7	6
FER-RATE	50	83	88	100
H24-2PN	4	5	1	5
H24-PS	0	0	1	1
H48-1C	0	0	0	0
H48-2C	4	3	1	0
H48-3C	0	0	1	0
H48-4C	1	3	4	4
PREG	Y	Y	Y	Y
OHSS	N	N	N	N
ABORT	N	Y	N	N
ECTOPIC	N	N	N	N

Table 4.2: *Retrieval function. Similar cases in the second stage of prediction.*

discovered, and the case base can be structured into clusters of relevant cases.

In $\mathcal{TA}3_{IVF}$, the user poses a focused query, such as "What have the pregnant women in common?", and the system finds a context which makes a given class of cases similar. Thus, $\mathcal{TA}3_{IVF}$ finds salient attributes and values for these attributes, i.e., a generalized interpretation of a given set of cases. This process is user-directed, interactive and possibly iterative. The user specifies the query, which constrains the search space, and the threshold T, which represents the certainty factor. The initial query is evaluated to produce a subset of the case base (e.g., find cases where $PREG : Y$ and $ABORTION : N$). Then, cases within the selected group are analyzed to produce a case interpretation that guarantees the coverage T (e.g., at least T% of cases have protocol $PROT1 : 1, 2$). The result is confirmed by an alternative query (e.g.. for cases where $PREG : Y$ and $ABORTION : Y$ determine $PROT1 : 1, 2$).

The coverage gives us the level of uncertainty for a particular attribute. It should be noted that the algorithm also confirms the result by evaluating alternative queries, such as pregnant patients with abortion. Only then could the result be selected as a good predictor.

In Chapter 6 we show how user-guided knowledge mining can be used to discover domain knowledge. In turn, this knowledge is used to: (1) organize attributes into categories, (2) to help in specifying the context, and (3) to help in generalizing or specializing attribute value constraints.

4.6 Discussion

This section showed how variable-context similarity assessment can be applied to support flexible case retrieval and user-guided knowledge mining. The next chapter describes implementation of the $\mathcal{TA}3$ system, and algorithmic evaluation of the underlying algorithms. Chapter 6 includes competence and efficiency evaluation of the prototype on diverse domains and various tasks. In (Jurisica and Nixon, 1998) we present a goal-oriented, knowledge-based approach for aiding development and usage of $\mathcal{TA}3$ decision support system for a medical domain. We propose an approach for dealing with non-functional requirements for CBR systems. We show how quality can be built into a CBR system, using the "QualityCBR" approach, which integrates existing work on CBR and non-functional requirements (Chung et al., 1998). We illustrate the use of the approach on an *in vitro* fertilization domain. The QualityCBR approach is used to address important non-functional requirements, such as performance, accuracy and confidentiality.

Chapter 5

The $\mathcal{TA}3$ System

This chapter describes $\mathcal{TA}3$'s main components, functional specification, and discusses case base management issues. After introducing the overall architecture of the system, we present modules for similarity-based retrieval, knowledge mining, case base organization, learning, and adaptation. When designing the system, our goal was a competent, scalable and flexible CBR system. To achieve this goal, we adopted an incremental view maintenance algorithm from database management systems, and used it for efficient context transformations during iterative browsing. We show that this yields a significant efficiency improvement for an interesting class of applications.

5.1 General Architecture

Given the theory proposed in Chapter 3, we have implemented a prototype CBR system $\mathcal{TA}3$. $\mathcal{TA}3$ stands for *TheAdvisor* 3.[1] Number 3 stands for the three major parts of *TheAdvisor* - representation, reasoning and presentation.

$\mathcal{TA}3$ comprises all modules of a case-based reasoning system: case representation, case retrieval, evaluation, adaptation and presentation. In addition, the explain function can be used for case base organization and knowledge mining. It is assumed that the case base is generated using external sources (we have used diverse methods for generating case bases in the performance evaluation studies presented in Chapter 6). Figure 5.1 depicts the overall architecture, data and control flow.

A *case base* is a persistent storage for cases. We extend a generic case representation described in Chapter 3 by organizing attributes that describe cases into one or more *categories*. Category membership is determined using information about the usefulness of individual attributes and their

[1] People who speak Slovak can appreciate a hidden meaning of $\mathcal{TA}3$ which phonetically stands for the highest mountain range in Slovakia - Tatry.

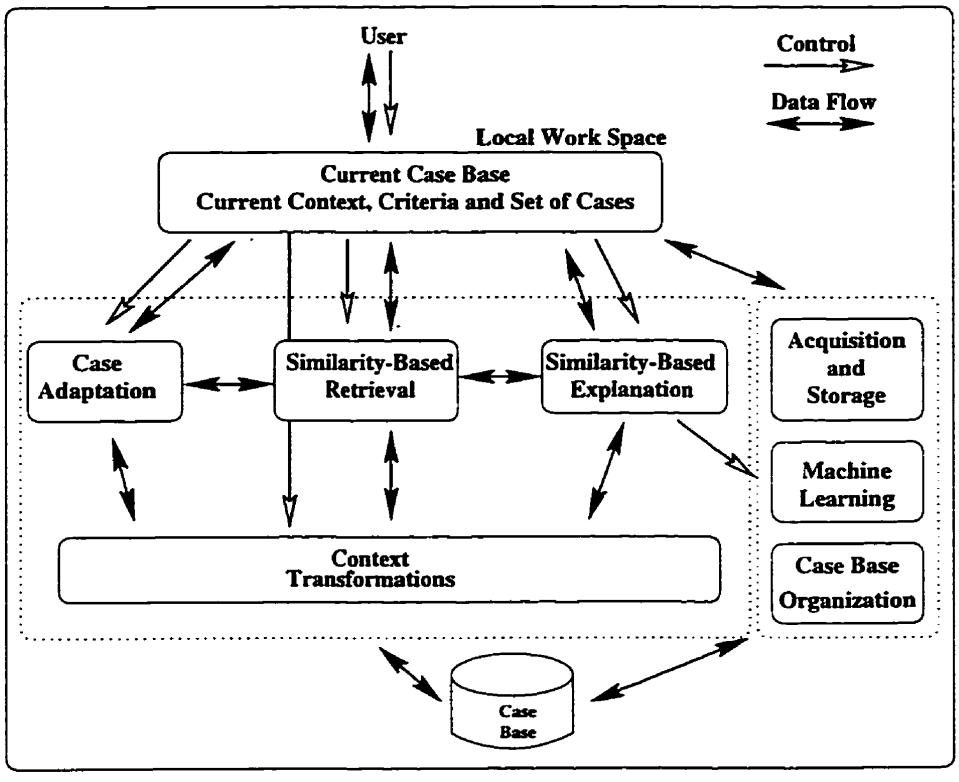


Figure 5.1: *TA3 architecture*

properties,[2]. Categories bring additional structure to a case representation. This reduces the impact of irrelevant attributes on system competence by selectively using individual categories during matching (Aha, 1992a; Ortega, 1995). A *context* explicitly defines attributes that are used during similarity assessment and any constraints that may be applicable to attribute values.

A set of cases represents returned cases that satisfy a given context during retrieval, or it may specify a subset of the whole case base during knowledge mining and case base organization. *Criteria* specify how the context should be handled during the retrieval and/or explanation process.. i.e., what context transformation should be applied. *Local workspace* serves as a temporary storage for cases, context and criteria.

The system supports the similarity-based case retrieval and similarity-based explanation (see Sections 3.5.2 and 3.5.3), context transformation functions, adaptation and knowledge management. User interface commands are for both functions presented in Section 5.3.

Depending on the application domain, the task at hand and availability of domain knowledge for adaptation *TA3* may operate in one of the three modes:[3]

1. **Case retrieval:** Case retrieval function is used to locate all cases relevant to a given context.

[2]This information is obtained either from domain knowledge or with help of a knowledge-mining algorithm.
[3]Chapter 6 describes application of *TA3* to problem solving in diverse domains.

All retrieved cases are presented to the user. It should be noted that the query could be subsequently modified using relaxation or restriction function. This process is described in Section 6.2.6, an application for supporting retrieval from a software repository.

2. **Case-based reasoning:** First, the case retrieval function returns cases relevant with respect to a given context. Second, case adaptation is used to construct a solution for the current problem, using attribute values from relevant cases. This process is described in, for example, Section 6.2.2.

3. **Knowledge mining:** Based on a given context, the system locates relevant cases and analyses their attributes and values, to help in finding regularities and patterns in data. This process is described in Section 6.2.1.

As described in Chapter 3, the context (query) can be modified in order to control quality (e.g., precision during case-based classification) and quantity (e.g., number of retrieved cases during retrieval from a software repository) by relaxing or restricting constraints on attribute values. Such transformation can be performed automatically, using background knowledge: context can be relaxed if fewer than specified number of cases is returned, or it can be restricted if more than a specified number of cases is returned. Because individual attributes in a case description may be placed into separate categories, the system modifies only one category at a time, namely, the least important category is relaxed/restricted first.[4]

TA3 is a decision support system, and thus it works best when used interactively, as a conversational case-based reasoning system (Aha and Breslow, 1997). Although a context can be transformed automatically, the user has the possibility to affect the change by manually restricting/relaxing individual attributes, or by selecting which attributes should be modified next. Thus, the user may control whether a context is transformed, and if it is, then how.

5.2 Case Representation

The purpose of this section is to discuss case representation used in *TA3*. As presented in Chapter 2.1, a case represents a specific knowledge (experience or episode) in a particular situation (context). Depending on the application domain and the CBR system, cases may have different form. Namely, they may differ in representation used, they may have various sizes, and they can be organized in diverse ways. Some applications require that all cases use the same attributes; however, in some applications different cases may use only subset of all possible attributes in a particular domain. Naturally, this requires using different representation formalisms. Usually, there is a tradeoff between expressibility and efficiency. It depends on a particular application domain and the task

[4] An arbitrary category is chosen if no ordering of categories is available.

under consideration, which tradeoff should be preferred. In Chapter 3 we have provided a theoretical basis for a flexible CBR system. The theory enables addressing the tradeoff, as will be exemplified in Chapter 6 on diverse application domains and various problem solving tasks.

Cases represent experience, usually in the form of problem-solution pairs. CBR systems are based on a process of remembering and recalling experience stored in a case base, namely cases that share similarity with a problem at hand. Because experience occurs in a particular situation that may differ from the current one, it is necessary to construct a mapping between past and present contextual information, as well as between the content of the case. Thus, cases represent the following information: (1) Context in which the experience was observed (problem description): the goal or task of the experience, situation description and constraints. (2) Content of the case (i.e., the actual experience): an action taken in order to solve the problem (solution and possibly reasoning steps, justifications, alternative solutions, expected outcome), an outcome of problem solving (a feedback and possibly explanation of the expectation failure, repair strategy, alternative solutions used). It should be noted that it is likely that not all cases will store knowledge with the same detail – some information may be missing, or it may not be useful in a particular situation.

A case may record experiences that are different from what is expected (Kolodner, 1993). However, we believe that even though this might be true for many CBR systems, there are domains where cases should represent experience in general. This way, the CBR technology could be used as a repository for corporate (or domain) knowledge. As will be shown in Section 6.2.1, we can use both positive and negative experience to generate a solution for a given problem. In addition, we can use deviation from expectation during knowledge discovery. For example, we show that after finding two patients with similar description, yet different outcome of treatment, it is possible to extend the schema to use additional descriptors that differentiate these patients.

Because individual application domains pose diverse requirements, case representation formalisms vary among CBR systems. The research so far tried to accommodate existing knowledge representation formalisms (e.g., predicate notation, rules, semantic networks or frames). As described in Chapter 3, *TA3* uses an object-based representation. Cases are stored as attribute-value pairs and are organized into categories. Category membership is determined using information about the usefulness of individual attributes and their properties. There are various approaches to determine, which attributes should be used. In the current prototype, we use two approaches – information is obtained either from domain knowledge or with help of a knowledge-mining algorithm (explain function). Categories bring additional structure to a case representation. This reduces the impact of irrelevant attributes on system competence by selectively using individual categories during matching (Aha, 1992a; Ortega, 1995).

$category_1$

$attribute_1$ value

$\vdots$

$attribute_n$ value

$\vdots$

$category_m$

$attribute_1$ value

$\vdots$

$attribute_k$ value

An example of a partial case in a medical domain is presented bellow. It shows that a case may have several categories and each category may comprise several attributes.

Patient_information		
	Patient_number	764
	Age	30
Diagnosis		
	Previous_diagnosis	tubal, MF
	First_diagnosis	1
	Second_diagnosis	3
IVF results		
	Pregnancy	Yes
	Abortion	No
	OHSS	No
	Ectopic_pregnancy	No

During problem solving, either a complete case representation may be needed or only portion of it may be relevant. In the latter situation, it may be inefficient to use all information stored in a case, but it can also be harmful, since it may lead to recalling irrelevant cases. Thus, it is more effective to access only relevant parts of the case. We use context to define relevant attributes and constraints on their values. Given the case defined above, one can specify the context as follows:

Patient_information		
	Patient_number	$D_{Patient\_number}$
	Age	$\{\langle 28, 32 \rangle\}$
Diagnosis		
	Previous_diagnosis	{tubal, MF, endo}
	First_diagnosis	{1}

In some domains, the system might reuse different parts of cases to produce a solution. All this can be characterized as a problem of case granularity, e.g., the size of knowledge pieces to be represented as a case and the organization of individual cases. There are two basic options:

1. Represent the whole experience as one case. The system should allow to access/reuse both the whole case and its portions (Jurisica et al., 1998).

2. Divide the whole experience into smaller chunks of knowledge and represent them separately. The system should, however, keep information about these pieces to allow for efficient access to the whole experience by connecting relevant pieces (Jurisica and Gupta, 1997).

In some domains, the decision about the case granularity is easy, e.g., for medical diagnosis, each patient could represent a case (see Section 6.2.1). It is a relatively small chunk of knowledge, independent of the other cases. However, particular tasks require more complex representation, where each case is composed of several knowledge parts (Jurisica and Gupta, 1997). Thus, the representation cannot be uniform for all cases.

5.3 Functional Specification

This section introduces functions that are supported in the TA3 system. All basic functions can be classified as one of the three types: These include:

- Reasoning-oriented functions, which include `retrieve`, `explain`, and `adapt`. Individual functions are selected on the basis of domain characteristics and a given task. Retrieve and explain functions are implemented using algorithms presented in Sections 3.5.2 and 3.5.3.

- Context manipulation functions comprise functions defined in Section 3.3, namely `reduction`, `expansion`, `generalization`, and `specialization`.

- Input/output functions, which are used to import cases, set criteria and present information to the user.

Additional functions, such as case base management, are defined using these basic functions.

5.3.1 Context Manipulation

Context manipulation functions include context transformations defined in Section 3.3. Context relaxation is implemented as reduction and generalization, and context restriction is implemented as expansion and specialization. Their application is either automatic or initiated by the user. The user controls automatic adaptation by specifying lower and upper limit on the number of retrieved cases, by defining which transformations can be applied, and which categories can be used (as defined

in *Criteria*). If the user does not specify the category that should be used, the system applies preferences specified in criteria.

`reduce`$(Context, Category) \rightarrow Context'$; `generalize`$(Context, Attribute) \rightarrow Context'$

`expand`$(Context, Category) \rightarrow Context'$; `specialize`$(Context, Attribute) \rightarrow Context'$

Reduction removes constraints by reducing the number of attributes required to match. Generalization relaxes the context by enlarging the set of allowable values for an attribute. As mentioned earlier, reduction can be performed automatically, taking advantage of attributes grouped into categories. A priority is assigned to each category (0 is the highest priority) to allow for more reliable relaxation algorithm. The priority is used during a selection process, to help decide which attributes are the most important ones and should be treated first. If no priority is assigned to attributes, they are treated in sequential order starting from the first one.

Generalization can also be performed automatically, assuming that domain knowledge is provided (generalization also uses category priorities, if no attribute is specified for generalization). In both situations, user can modify the context manually.[5] As proven in Theorem 3.6.2, the process of case retrieval is monotonic.

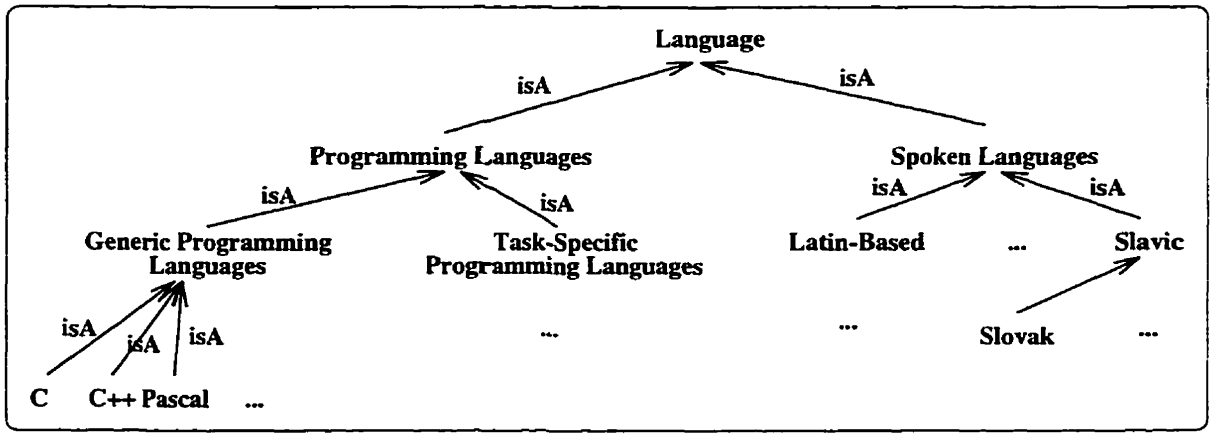


Figure 5.2: *Generalization and specialization along the is-a hierarchy.*

5.3.2 Similarity-Based Retrieval

The retrieval component of the *TA3* system aids the process of recalling cases from a case base by handling user queries flexibly. `retrieve` is a case retrieval function (defined in Section 3.5.2), which

[5] If available, is-a hierarchies (e.g., as in Figure 5.2) may be used to provide domain knowledge for application of generalization.

returns a set of cases from a case base that are relevant to a query with respect to a given context.

$$\texttt{retrieveIterative}(Context, CaseBase, lowerLimit, upperLimit) \rightarrow Answer$$

Criteria specifies how the context should be handled, i.e., how the context should be transformed for controlling the closeness of retrieved cases and the amount of cases that is returned (see Section 3.3 for description of context transformations).

If too many or too few relevant cases are retrieved using the initial query, then the system automatically transforms the context or a user may modify it manually. There are two possible transformations: relaxation (introduced in Definition 3.3.1) - to allow for retrieving more cases and restriction (introduced in Definition 3.3.2) - to allow for retrieving fewer cases. These context transformations are a foundation for supporting iterative retrieval and browsing.[6] Figure 3.7 defines an iterative retrieval algorithm. The `retrieve` (*Context*, *CaseBase*) function is based on the algorithm defined in Figure 3.6. Context specifies which categories and attributes are considered during matching, and what constraints are defined on attribute values.

In summary, three classes of functions are used during retrieval: (1) retrieval-oriented - used to retrieve relevant cases; (2) context-related - used to modify a give context; and (3) criteria-related functions - used to select an appropriate context transformation strategy. As was described above, retrieval is initiated by invocation of retrieve function. Initial context is defined by directly transforming a problem case into a context, as specified in Section 3.2.2. The user may, however, subsequently modify the initial context.

5.3.3 Similarity-Based Explanation

The similarity-based explanation component of the TA3 system aids a user in a knowledge-mining process by using alternative contexts that explain the selected set of cases. Given a case base and a set of cases the function returns a context that specifies constraints on attributes such that all cases in Set of cases satisfy them. Explain function can be used to find useful features for explaining a given set of cases (see Section 6.2.1) and thus it may help to create an efficient knowledge organization (Jurisica and Glasgow, 1997). It can also suggest a useful schema evolution (Jurisica et al., 1998). Explan function finds a context that uses the maximum number of attributes with the minimum number of constraints on them. The algorithm is presented in Figure 3.9.

`Explain` is the function that returns a context in which all cases specified in a set of cases can be considered similar. Criteria specifies how the context should be handled, i.e., how the context should be transformed for controlling the closeness of retrieved cases (see Section 3.3 for description of context transformations).

[6] In Section 5.4 we present an incremental algorithm that supports efficient iterative browsing.

The explain function (see Section 3.5.3) is similar to knowledge mining in databases, namely mining for classification rules (Ramakrishnan, 1998). The general principle is to find appropriate values for attributes $a_1, ..., a_k$ that could be used to predict (or classify) the value of attribute a_p:

$$(ll_1 \leq V_1 \leq ul_1) \wedge ... \wedge (ll_k \leq V_k \leq ul_k) \rightarrow (a_p : V_p),$$

where *ll* is the lower limit for attribute value and *ul* is the upper limit. The algorithm uses principles similar to the *CDP* algorithm (Agrawal, Imielinski and Swami, 1993).

In *TA3* , the user poses a focused query, such as "What have the pregnant women in common?" and the system finds a context, which makes a given class of cases similar (see Section 6.2.1 for more detail). In other words, *TA3* finds salient attributes and values for these attributes, i.e., a generalized interpretation of a given set of cases. This process is user-directed, interactive and possibly iterative. The user specifies the query that constrains the search space and the threshold T, which represents the certainty factor.

The main idea of the knowledge-mining algorithm is to identify attributes with similar characteristics. We use variability of attribute values to determine which attributes are likely to be good predictors, i.e., it is a characteristic feature. If there is a high variability of values for a particular attribute, then the attribute is not a good characteristic feature. Low variability of attribute values suggests a characteristic feature. When all attributes are processed, the system groups similar attributes into categories, i.e., characteristic features are separated from other attributes. Once cases are described by their characteristic features, the case base is dynamically organized into particular clusters, defined by contexts.

This approach to attribute-oriented knowledge mining is similar to factor analysis (Harman, 1976). Factor analysis is a statistical technique used to analyze interrelationships among a variables, and to explain these variables in terms of their common factors. Factors group correlated variables that are largely independent of other subsets of variables. Thus, it is similar to grouping related attributes into categories.

In a summary, explain function finds *typical* case for a given group of cases, i.e., it is the most typical (or frequent) set of attribute-values. If this typical case is used as a context, then all cases similar to it form an equivalence class. A *prototype* may be different from typical case; it is a representative case, which may or may not be a typical case. The function also supports schema evolution. After two cases are found (logically) equivalent, but with different outcomes, (1) new attributes may be added to distinguish these cases, (2) the two cases may be grouped into a cluster (an equivalence class), or (3) only one of the cases is stored, but the strength of the experience is increased.

Explain function can be used to discover attributes that are important during case retrieval. This

may improve both competence and scalability of a case-based classification system, due to reducing the effect of irrelevant attributes on classification accuracy (Jurisica and Glasgow, 1997). Another important benefit of using explain function is discovering proper domain representation. Namely, by locating cases that appear to be similar, yet their classification outcome is different, one can extend the representation of cases to include new attributes that help differentiating these cases (Jurisica et al., 1998). We have evaluated the effect of explain function on the TA3 performance on several domains, which is reported in Chapter 6.

In summary, similarity-based explanation uses explain function, context-related and criteria-based functions. The user modifies system's behavior by specifying threshold for explain function and by selecting set of cases for consideration.

5.3.4 Case Adaptation

Adapt function can be applied in some domains to compute attribute values for a returned case on the basis of the attribute values in the set of retrieved cases (as explained in Chapters 4 and 6).[7] In the current implementation, only simple adaptations are supported, namely, numeric attributes can be adapted by selective averaging of attribute values.

$$\texttt{adapt}(SetOfCases, Context) \rightarrow Case$$

Averaging of attribute values is selective because retrieved cases are ordered according to the class and then according to the relevance measure, as described in Chapter 4. Thus, during adaptation the preference is given to positive examples that closely match a given case. If a domain knowledge is available in the form of is-a hierarchies and/or specific adaptation rules, additional case adaptation can be provided.

5.3.5 Case Presentation

Input/output functions are used to import cases from external files and display context, criteria, case base, individual cases, and set of retrieved cases. Examples of the user interface are presented in Chapter 4.

5.4 Incremental Context Modifications

The relaxation technique can advantageously be used for returning answers to a specific query as well as returning related answers (Gaasterland, 1993). Without an automatic query relaxation, users would need to submit alternative queries. The restriction technique works analogously, but is

[7] It should be noted that domain knowledge and user intervention is used during the adaptation process.

used mainly for controlling the amount of returned information, preventing information overload. Since the search for relaxed or restricted contexts could be infinite, there must be a mechanism for controlling it, either by user intervention (via user preferences) or by other means.

A context-transformation process (iterative restrictions or relaxations of a context) produces a chain of successive contexts $(Context_0, Context_1, \cdots, Context_n)$ and corresponding sets of retrieved cases $(Answer_0, Answer_1, \cdots, Answer_n)$. This process is monotonic in the sense that if the successive contexts are partially ordered then the retrieve function produces a corresponding effect on the sets of cases retrieved. More formally, if $Answer_i$ is the set of cases retrieved for context $Context_i$, then a partial order (by relaxation) of contexts results in a partial order (by subset) of case sets: If $Context_0 \prec Context_1 \prec \cdots \prec Context_n$, then $Answer_0 \subseteq Answer_1 \subseteq \cdots \subseteq Answer_n$.

A context can be similarly iteratively restricted by making it progressively more specific, i.e., allowing fewer cases to satisfy it. As in the relaxation algorithm, expansion and specialization algorithms produce a partial order of contexts. In general, only one category is restricted at a time, using categories with the lower priority first.

The process of restricting and relaxing contexts can be repeated and interwoven until the agent is satisfied with the quantity and the relevancy of the retrieved cases. It is apparent that after the context is modified by relaxation/restriction, the system must re-evaluate the query. A naive approach takes the new query and submits it to the system. A more sophisticated approach could take advantage of an already processed query by incrementally modifying its result (Bancilhon, 1986). In the next section, we introduce such an algorithm.

There are many applications where iterative browsing is an effective way of obtaining information (Constantopoulos and Pataki, 1992; Martin, Hung and Walmsley, 1992). It is particularly useful in complex domains for exploratory search, and for exploratory problem solving (i.e., examining "what-if" scenarios). Such applications include data mining problems (Brachman et al., 1993), online analytical processing, CBR (Leake, 1996), etc. In order to support iterative browsing, a system must be able to respond to a series of queries, where the system or the user may alter a current query and re-submit it for further evaluation. This process may be repeated until the desired quality and quantity of information is obtained. The iterative process is aimed at maintaining high recall while improving precision.

A naive approach to iterative browsing involves evaluating each query independently. A more sophisticated approach involves incremental computation (Ceri and Widom, 1991; Griffin and Libkin, 1995; Gupta, Mumick and Ross, 1995), where the result of a query is reused to evaluate a subsequent query more efficiently. The amount of necessary modification to a query in an incremental approach can be, to some degree, controlled by collecting and using extra information produced during query evaluation. Although this may require additional storage, an overall efficiency is usually improved.

We propose an incremental retrieval algorithm for CBR systems, which is based on a nearest-

neighbor matching algorithm (Wettschereck and Dietterich, 1995). We modified it to include:

1. grouping attributes into categories of different importance to gain a fine-grain control over the matching process, and diminish the negative effect of irrelevant attributes on competence;
2. using an explicit context during similarity assessment to recall only relevant cases; and
3. accelerating query processing via incremental context transformation during query relaxation.

5.4.1 Relevant Research

Browsing requires that the query (and the context) be iteratively changed, by restricting or relaxing it. The process of restricting and relaxing contexts can be repeated and interwoven until the agent is satisfied with the number and the relevancy of the retrieved cases. It is apparent that after the context is changed by relaxation/restriction, the system must re-evaluate the query. A naive approach would take the new query and would submit it to the system. A more sophisticated approach can take advantage of already processed query by incrementally modifying its result (Bancilhon, 1986).

Incremental view maintenance algorithms have been successfully applied to database systems (Bækgaard and Mark, 1995; Ceri and Widom, 1991; Griffin and Libkin, 1995; Gupta, Mumick and Ross, 1995). Various approaches have been proposed to handle view deletions, negations, updates, aggregation and recursion. One well-known incremental view maintenance algorithm is a counting algorithm (Gupta, Mumick and Subrahmanian, 1993), which supports delete and re-derive operations. It assumes universal materialization of predicates and stores counts of the number of derivations to be associated with each tuple. Ceri and Widom propose an algorithm for deriving production rules to maintain selected SQL views, namely views without duplicates, aggregation and negation is proposed (Ceri and Widom, 1991).

Incremental view maintenance algorithms avoid a complete query re-computation by changing only relevant parts of the answer or view. Usually, an incremental approach is substantially more efficient than a naive one. Efficiency improvement is higher when several consecutive changes to the query are required (i.e., during iterative browsing), or when a large information base is used. Small updates to the query generally produce only small changes to the query result. Then an incremental approach performs only local changes to the query (Gupta, Mumick and Ross, 1995).

Gupta, Mummick and Ross (1995) present an incremental algorithm for adapting the view in response to changes in the view definition. The authors consider an SQL *Select-From-Where-GroupBy*, *Union* and *Except* views, and present local adaptation strategies using the old view materialization. Their methods for adapting the *Where* part of an SQL view are similar to our context modifications. However, they do not support cardinality relaxation and restriction. Blakeley, Larson and Tompa introduced an incremental view maintenance algorithm that supports only base relations updates (Blakeley, Larson and Tompa, 1986). In contrast, *TA3* also supports updates to context (i.e., views).

FRANK is a case retrieval system, applied in a medical domain for back-injury diagnosis (Rissland et al., 1993). The user provides a description of a patient's symptoms and selects from a hierarchy of report types. The user's top-level considerations are filtered through the system's processing using a flexible control mechanism. The plan is then selected based on the report type. The task mechanism controls queries to the case base – if a query is not successful, then the system resubmits the query with altered initial values. Thus, it is similar to our notion of iterative query modification.

5.4.2 Incremental Context Modification in TA3

The basic idea of incremental query processing is to store query results and reuse them to compute related queries (Bækgaard and Mark, 1995). If the number of attributes per case is significantly smaller than the total number of cases in the case base then incremental context modification outperforms computation of the answer from scratch. Our system supports incremental context modification when partial results are kept at the attribute or category level. The first approach requires extra storage space, but is more versatile and is thus suitable when context is changed frequently and when cases have fewer attributes. The second approach is a compromise between efficiency gain from an incremental approach and modest storage requirement. It is useful for less frequent context changes and for case bases containing cases with a large number of attributes. Next, we explain the rationale behind the first approach. (The second approach works analogously.)

Consider context relaxation with the naive and incremental retrieval algorithms (see Figures 3.7 and 5.3). During k iterative context modifications the naive case retrieval algorithm requires k iterations, while the incremental algorithm handles it with only a single iteration. Although the initial evaluation is the same as for `retrieveNaive`, the incremental approach stores partial matching results on the attribute level in the set of cases $Answer_i$. If a case is a member of $Answer_i$ it means that it satisfies the constraint on $attribute_i$.

During iterative retrieval of cases, the system modifies constraints on attributes. A *naive retrieval algorithm* (see Figure 3.7) produces $Answer'$ by determining which cases in a case base satisfy constraints on all attributes defined in the context, i.e., for all $Answer.attribute_i$. *Incremental retrieval algorithm* reuses $Answer.attribute_i$ to produce the set of relevant cases $Answer'$ (see Figure 5.3).

Relaxing attribute $attribute_i$ changes a set of matching cases included in $Answer.attribute_i$. However, all remaining partial answers remain unchanged. Thus, $Answer'$ can be constructed by adding cases that satisfy both the initial context and the relaxed constraint on attribute $attribute_i$ to the $Answer$. Restricting attribute $attribute_i$ results creating $Answer'$ by removing cases that do not satisfy additional constraint from $Answer$. Determining satisfiability only requires testing if the case in the set of retrieved cases must be removed either because it needs excluded value to match or because it cannot match an added attribute.

The idea of incremental context relaxation and restriction has evolved from the notion of differ-

ential queries (Blakeley, Larson and Tompa, 1986). First, it is determined what parts of the context are affected by the proposed change. Second, only those parts are recomputed. We express this process using context addition and difference. Thus, the incremental context transformation is as follows: $Context' = Context + \delta^+ - \delta^-$, where δ^+ and δ^- denote a context that needs to be added or removed.

Without loss of generality, it is assumed that δ^+ and δ^- are contexts with single attribute/constraint pair. This is sufficient, since the context transformation process is iterative and thus more complex constraints can be created iteratively. Next we formalize context transformations for our incremental retrieval algorithm and show how the final set of retrieved cases ($Answer'$) can be constructed using partial results (on attribute level) from previous query evaluation ($Answer_i$).

Reduction. *Reduction* involves removing an attribute-value pair from a context. This can be done either *permanently* – an attribute $attribute_k$ is removed from the context: $Context' = Context - attribute_k$, or *dynamically* – an *m-of-n* matching is used. Thus, for the reduced context, the resulting set of cases is generated as a union of the partial results of the set of cases that satisfy individual attribute constraints of the original context ($Answer.attribute_i$), without considering the constraints on the removed attribute: $Answer' = \cap Answer.attribute_i / Answer.attribute_k$.

Expansion. Analogically, *expansion* involves adding an attribute-value pair to a context: $Context' = Context + \delta^+$, where $\delta^+ = \{attribute_k.\{Value_k\}\}$. Thus, removing cases that do not satisfy the constrained context from the set of cases generates the set of retrieved cases: $Answer' = Answer \cap Answer.attribute'_i$, where $Case_i \in Answer.attribute'_i$ *iff* $sat(Case_i, \delta^+)$.

Generalization. *Generalization* involves enlarging the set of allowable values for a given attribute in the context: $Context' = Context + \delta^+$, where $\delta^+ = \{attribute_k.\{Value_k\}\}$. Thus, the set of retrieved cases is generated as an intersection of the set of cases that satisfy the original context and the set of cases that satisfy the context change δ^+: $Answer' = Answer \cap Answer.attribute'_i$, where $Case_i \in Answer.attribute'_i$ *iff* $sat(Case_i, \delta^+)$.

Specialization. *Specialization* involves removing values from a constraint set for an attribute defined in a context: $Context' = Context - \delta^-$, where $\delta^- = \{attribute_k.\{Value_k\}\}$. Thus, the set of retrieved cases ($Answer'$) is generated by removing cases that do not satisfy the restricted context from $Answer$: $Answer' = Answer \cap Answer.attribute'_i$, where $Case_i \in Answer.attribute'_i$ *iff* $sat(Case_i, \delta^-)$. such that: $sat(Case_i, \delta^-)$.

```
retrieveIncremental (Answer, Context, CaseBase, LowerLimit, UpperLimit)
    for all cases in the CaseBase
        if a case Case_j satisfies constraints on attribute_i then
            add (Case_j, Answer_i)
    for all attributes in the Context
        Answer = ⋂ Answer_i
    if consecutive relaxation-restriction or restriction-relaxation then
        return (Answer)
    else if | Answer | ≤ LowerLimit then
        relax (Context, Category)
        set Answer.attribute'_k to cases that satisfy the relaxed category
            if reduction then
                Answer' = ∩ Answer.attribute_i/Answer.attribute_k
            else if generalization then
                Answer' = Answer ∩ Answer.attribute'_k
    else if | Answer |> UpperLimit then
        if the Context was not previously relaxed then
            restrict (Context, Category)
        set Answer.attribute'_k to cases that satisfy the relaxed category
        Answer' = Answer ∩ Answer.attribute'_k
    retrieveIncremental (Answer', Context, CaseBase, LowerLimit, UpperLimit)
end
```

Figure 5.3: *Incremental case retrieval algorithm. Context is initialized with the attributes and constraints from the input case. Special counters are used to prevent repeating context restrictions and relaxations forever. Context transformations modify attributes of the least important category first.*

5.5 Knowledge Management in the TA3 System

The purpose of this section is to discuss the adaptive and learning components of the TA3 system. According to (Simon, 1983) learning denotes changes in the system that are adaptive in the sense that they enable the system to do the same task or tasks drawn from the same population more efficiently and more effectively the next time. This might require not only new knowledge, but also increasing the effectiveness with which the system uses the knowledge it already possesses. Two distinct research areas have been developed: (1) acceleration of problem solvers, and (2) inducing concepts from examples. Machine learning may be used to improve system performance: either by extending the space of possible solutions (improving system's competence) or by shortening the time required to produce an answer (scalability). However, nothing comes free, and a machine-learning algorithm may reduce system scalability because of the added computational complexity, i.e., it will suffer from a utility problem (Greiner and Jurisica, 1992; Mooney, 1989). In other words, there is a tradeoff between advantage of storing more cases in a case base, and facing a a more complex retrieval of relevant cases (further detail on a utility problem are provided in Section 5.5.1). Next we discuss how machine-learning approaches can effectively be used to improve CBR systems, to improve case acquisition, case base organization and case-based reasoning. We show how TA3's

approach to case representation and explanation function support case base management.

5.5.1 Knowledge Acquisition

Knowledge acquisition can be considered a simple form of learning - accumulating experience over time. Knowledge acquisition in case-based reasoning is usually an incremental process, new cases are added into an existing case base. There are several issues that need to be considered, namely a case base construction and an optimal size of a case base.

In the CBR paradigm the distribution of cases in the problem space and their form affects a system's performance (both competence and efficiency). The system requires more specific cases to cover the same problem space than it would require generic cases. Obviously, there is a tradeoff between competence and scalability. The larger the case base, the more problems can be solved. However, the larger case base results in an increased computational complexity of the case retrieval and adaptation algorithms.

Form of a Case Base

An important issue, related to representation, is case base construction. As was already mentioned, it is not a trivial task to decide how to represent a case in a particular domain and what cases are worth storing because there is a tradeoff between added competence from new cases and higher cost of case retrieval (Minton, 1988a; Hunter, 1989; Francis and Ram, 1993).

Moreover, the required case representation can be complex, which makes it difficult to acquire a sufficient number of cases for a particular application. These problems are the main reasons why many CBR systems have relatively small case bases.

The process of a case base construction can be simplified using the following approaches. A case base is created automatically from an existing database. An example of this approach is the FGP Machine (Fetch, Generalize, and Project) (Fertig and Gelertner, 1991). The FGP Machine is a software architecture that processes large databases[8] and makes the knowledge from them explicit. We have used a similar approach in our IVF application (see Section 6.2.1).

Another approach is to use a special intelligent system as an assistant to case acquisition. A representative of this approach is the Case-Based Formulation system (Branting, 1992). The system is based on the use of previous cases as a model and guide for expressing new cases. A copy-and-edit approach is used, where a frame is added to a case base by finding a similar frame, copying it, and modifying the copy. This process can use either single case or multiple cases. The system allows a goal specification in order to constrain the matching process; it is an interactive method, automatically insuring consistency with existing collection of related facts with a growing case base.

[8] However, the current implementation is inadequate to handle case databases of more than 1,000 cases assuming 10-20 features per case. The largest test-case-base consists of 643 cases, each having 3.6 features on average.

As will be described in Section 6.2.2, it is also possible to populate the case base automatically, for example, by using an underlying system (e.g., forward kinematics task solver, planning system). This approach is used to control the coverage of the problem space by cases in a case base.

TA3 also supports an artificial generation of a case base by adding hypothetical cases, as will be described in Section 6.2.4. This approach is used to extend the existing small case base. As will be shown, system competence can be improved, but the system must ensure that no incorrect problem-solution pairs are added to the case base.

The Utility Problem

It would appear that the more cases are stored in a case base, the better the system's competence. However, because of the utility problem (Minton, 1988b; Holder, 1992; Francis and Ram, 1993; Ram and Santamaría, 1993b), the system's efficiency may degrade with increased case base size. The reason is that after the case base becomes unmanageably large, it might be easier to solve the problem from scratch than to find a best match in a case base and adapt it for current needs.

One can solve this problem by: (1) limiting the number of remembered cases and (2) improving case storage and retrieval algorithms. Rarely all observed cases are worth remembering. The system should determine the effort needed to adapt the case and use it during judging if the case should be stored for later reuse or not. A more sophisticated method, based on the same principle, is referred to as exemplar-based reasoning, i.e., the system remembers only significant, representative cases – exemplars (Kibler and Aha, 1987; Bareiss, 1988; Branting, 1989; Porter, Bareiss and Holte, 1990; Aha, 1995). This has an advantage of not decreasing (or decreasing insignificantly) the competence, yet improving the scalability. Even higher level of "pruning" the number of observed cases is used when cases are stored only after generalized (Branting, 1989; Jones, 1991; Portinale, 1991)

Case base management systems cope with the utility problem by improving the algorithms for efficient storage and fast retrieval. This approach is inevitable in the areas where even storing the exemplars would cause efficiency degradation. Moreover, even for the exemplar-based systems, case base management can help improve system's scalability. This may be required in some applications, where resources are limited or where real-time performance is needed (Jurisica, 1996).

TA3 copes with the utility problem by (1) using variable-context similarity assessment, (2) using an efficient incremental retrieval algorithm, and (3) by producing approximate solutions quickly, and improving them if more computational resources are available. From the last viewpoint, **TA3** works as an anytime algorithm (Dean and Boddy, 1988).

5.5.2 Knowledge Evolution

The purpose of this section is to introduce a notion of knowledge evolution and discuss how is it supported in **TA3**. First, we motivate the need to support a knowledge evolution. Second, we

present mechanisms for CBR systems that support gradually-changing knowledge.

Knowledge evolves regardless of our wishes due to environment changes, emergence of new problems or modification of our goals, user model changes, and evolution of our understanding of the surrounding world by discovering new relationships and principles. Recalling that cases represent experience, we need to determine the difference between *a priori* and *a posteriori* knowledge. According to Kant, *a priori* knowledge is knowledge which exists independent of experience; it is not derived from experience, but from a general rule, which was created based on experience (Kant, 1947). A posteriori knowledge is defined as the knowledge in experience. Using Kant's view, cases represent a posteriori knowledge, while generalized cases represent *a priori* knowledge.

Kant also divides *a priori* knowledge into pure and impure, based on their independence of empirical knowledge. Pure *a priori* knowledge has no empirical element, e.g., the statement "Every change has a cause" is a proposition *a priori*, but impure, because change is a conception which can only be derived from experience.

Furthermore, knowledge is dynamic not static, which should be reflected by the representation formalism chosen. On the one hand, instances (or cases) tend to be an efficient representation formalism to capture ill-structured domains or knowledge in the early stages of acquisition (Brewer, 1989). On the other hand, other forms of knowledge tend to be represented in terms of abstract structures, such as rules or case generalizations (Brewer, 1989). It is also observed that the apparent changes with age in analogy use, problem solving, and transfer are due not to the development of ability in the child but to the acquisition of knowledge.

Analyzing existing CBR systems, it was observed that real-world cases are incomplete, large and complex, and that a successful system must support multiple types of knowledge (Pearce et al., 1992). It is likely that the case base is created over a period of time and that the case representation evolves, as the problem domain is better understood. Thus, the process of acquiring expertise using a CBR paradigm can be divided into two phases:

1. During the initial phase, experience is acquired and the first case representation schema is defined. Cases are stored in a case base, which is organized to support efficient access.

2. During the case base maturing phase, additional cases can be acquired, and the schema for case representation may change. Thus, the case base organization must evolve to accommodate new cases and changed schema. In addition, existing cases may be updated using reinforcement learning (based on how they are accessed and used).

TA3 supports knowledge evolution by using explain function. Interactive exploration of the domain may reveal that the case representation used is not sufficient to distinguish some cases, i.e., cases that have similar problem description but different solutions (or outcomes). This results in using additional attributes to describe cases, as was the case in IVF study (Jurisica et al., 1998).

Explain function can also be used to find structure in the case representation, i.e., find attributes that are relevant during problem solving, and their representative values. This can also lead to clustering a case base into equivalence classes, i.e., grouping together cases that are similar in a given context. Knowledge evolution is also supported by the mechanism of forgetting, which is described next.

5.5.3 Forgetting

Forgetting is a technique that is used to remove cases that neither improve solution accuracy nor extend the problem space. Obviously, candidates for forgetting must be selected according to certain measures, so the reasoning potential of the system is not decreased. There are two approaches to forgetting: (1) a particular case may be removed completely from the case base, or (2) a particular case may be stored into a secondary storage. Obviously, the first approach is more radical. The second approach improves the competence in most frequent situations, while rare situations are solved with extra processing required.

Several heuristics exist that can be used to decide which case is a candidate for removal from a case base. An obvious approach is that if a solution for a particular problem is easily adapted from already stored case, then the current case is only a supporting case and thus need not be stored. Another useful heuristic is based on the idea that if a case base is being developed over period of time and the task or context changed in the meantime, older cases may be removed, since they are not relevant under current situation (Widmer and Kubát, 1993).

Selective forgetting (Cox and Ram, 1992; Smyth and Keane, 1995; Watanabe, Okuda and Fujiwara, 1995; Widmer and Kubát, 1993) is one of the techniques used in CBR to cope with improving efficiency of the case retrieval algorithm without decreasing its competence.

Forgetting was not used in the current implementation of TA3 . It is feasible, though, to use a "soft forgetting" in the robotic domain (see Section 6.2.2) to store infrequently accessed cases in the secondary storage to speed up retrieval of frequent robot paths.

5.5.4 Learning

The obvious learning method in case-based reasoning is acquisition of new cases, which in machine-learning community is referred to as rote learning (or learning by being told). There are, however, other learning mechanisms that may be sometimes useful.

Knowledge discovery and inductive learning are used in CBR for supporting case base evolution and for finding relevant contexts. Such techniques can be used for *attribute optimization*, i.e., reducing or increasing number of attributes used for representing cases. This process results in schema change that may lead to better discrimination among similar cases and improved competence due to removal of irrelevant attributes. A related task is to find out which attributes are the most relevant when solving a particular task and to find representative values for these attributes. TA3

uses the explain function for attribute optimization.

Another fruitful application of knowledge discovery and inductive learning is *case base organization*: Efficiency of case retrieval and accuracy of case-based problem solving can be increased if relevant cases are grouped (clustered) together. TA3 uses the explain function to find cases similar in a given context.

After the case base is mature, i.e., the rate of new cases being acquired is stabilized, knowledge discovery and inductive learning algorithms can be used to generalize the knowledge contained in the case base. The generalization of cases can be used for better understanding of underlying principles in a particular domain and for more efficient organization of a case base into generalization hierarchies.

Because not always enough knowledge is available to form a case base with ideal distribution of cases (as can be seen in Figure 5.4), it is beneficial if the system can automatically create and evaluate such knowledge pieces. This process is called automatic generation of hypothetical cases. One of the possible ways to extend case base automatically is to deploy evolutionary learning, namely genetic algorithms. It should be noted that such a technique has a limited applicability, since many domains would not allow for such a possibility due to their complexity and variability. However, as will be shown in Section 6.2.4, in simpler and well-defined domains, applicability of evolutionary algorithms for case base enhancement is a viable possibility. In this domain, we showed how the accuracy of a case-based classification system may be improved using a cross-over function.[9]

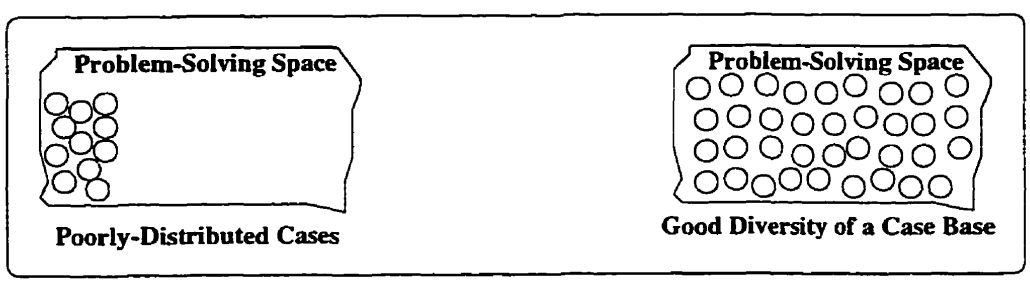


Figure 5.4: An effect of case diversity in a case base.

5.6 Chapter Summary

This chapter described how the theory presented in Chapter 3 has been implemented, and what functionality is available in the TA3 case-based reasoning system. In addition, we have shown how the flexible variable-context similarity and similarity-based retrieval algorithm can be extended using context transformations to support efficient iterative browsing and knowledge mining.

A similar approach to our variable-context similarity-based iterative retrieval algorithm is used in

[9] Other possible operators include deletion, insertion, inversion, translocation, mutation, selection, division and connection.

the FindMe system (Hammond, Burke and Schmitt, 1996). Here, the system supports and interactive refinement of a set of retrieved cases by adjusting feature importance. A work by Kolodner (1993) on situation assessment problem in CBR is relevant to our similarity-based explanation – it concerns selection of appropriate features that can be used in the query. The feature-selection problem is also tackled by Cardie for case-based learning of linguistic knowledge (Cardie, 1996). Leake's work on indexing explanations by expectation failures relates to our relaxation process an is applied when the initial query does not return a satisfying result (Leake, 1992a).

Efficiency of the incremental context-transformation algorithm is achieved by adapting an incremental view maintenance algorithm from database management systems. Performance of the algorithm is evaluated in Chapter 6 and Chapter 7. The former evaluates competence of the system, the latter evaluates its scalability.

Case base management issues have been discussed to relate the needs of CBR systems to knowledge base management efforts. We considered issues, such as case acquisition, case representation, case base organization and case representation (and knowledge) evolution. We showed how case representation used in **TA3** and similarity-based explanation can be used to satisfy case base management issues.

TA3 is implemented in C++ and has a text-based interface in the UNIX environment and a graphical user interface in the MS Windows environment (only for IVF domain). Although **TA3** is a generic system in the sense that it can be applied to different domains and it supports solving diverse tasks, some manual steps are required to prepare it for a particular domain. Namely, schema for representing cases must be encoded into the main header file and the system must be recompiled.

In the next chapter we provide further description of **TA3** and its performance evaluation (its competence and efficiency) on diverse problem domains. Further system description and outline of **TA3** deployment as a decision support system is discussed mainly in Sections 6.2.1 and 6.2.6. Appendix A.6 discusses suitable application areas for **TA3** and describes how it can be deployed.

Chapter 6

Performance Evaluation

This chapter discusses performance evaluation methods for CBR systems and assesses competence of $\mathcal{TA}$3 on several real-world domains. (Next chapter evaluates $\mathcal{TA}$3's efficiency.) There are many accepted benchmarks used in different fields. It is important to decide what to measure and how to interpret the results. We evaluate $\mathcal{TA}$3's performance and compare it to that of other systems. We consider both system competence (i.e., solution quality), and system efficiency and scalability

---

6.1 Evaluation Framework

In this section we present relevant benchmarks, discuss issues pertinent to evaluating system performance, and introduce task scenarios for assessing $\mathcal{TA}$3's competence and scalability.

6.1.1 Evaluation Measures

Precision and recall are used to evaluate information retrieval systems, while accuracy and coverage evaluate machine-learning algorithms. The former benchmarks are used to assess how good the retrieval is, i.e., how much relevant information is retrieved and how much irrelevant information is included. The latter benchmarks are used to measure accuracy of the learned rule and the portion of an information base it covers. Next, we discuss how these measures apply to CBR system evaluation.

Precision and Recall

The most accepted benchmarks for information retrieval systems are *recall* and *precision*. Precision can be explained as a measure of avoidance of irrelevant items. Thus, the higher the precision, the greater the percentage of relevant items in the set of retrieved items. Recall is a measure of completeness of retrieval. Thus, the higher the recall, the smaller the set of relevant items that

are not retrieved. Formally, precision and recall are given by: $Precision = 100 * RelRet/Ret$ and $Recall = 100 * RelRet/Rel$, where Ret is the number of retrieved items, Rel is the number of relevant items, and $RelRet$ is the number of relevant retrieved items,

Given the nature of information retrieval systems, when recall rises, generally precision decreases and vice versa. In contrast, variable-context similarity assessment helps to achieve both high precision and high recall. Next, we describe how this can be achieved.

Two iterative retrieval approaches yield high precision and recall simultaneously:

1. **Conservative approach:** The objective is to prevent retrieving irrelevant items, even if some relevant items are missed. After the initial set of relevant items is available, we could iteratively consider neighbors and include only relevant ones (so that precision is not decreased). This approach is appropriate if the user is knowledgeable in the problem domain, since the retrieval process must be constrained – to eliminate irrelevant matches, and similarity-based – to retrieve relevant neighbors of initial matches. The conservative approach can be explained as *precision-oriented retrieval* with iterative query tuning to increase recall, while preserving high precision.

2. **Novice approach:** The emphasis is on retrieving all relevant items, even at the price of including some irrelevant ones. After the initial pool of items is retrieved, irrelevant items are iteratively removed. This approach is suitable for novices, i.e., users who do not have initial knowledge about the structure of the repository. Thus, they may not specify a query that ensures high precision, but using the result and feedback information they may iteratively prune the set of retrieved items. The novice approach can be explained as a *recall-oriented retrieval* with iterative query tuning to increase precision, while preserving perfect recall.

From the above description it follows that flexible criteria are required to guide and control retrieval. Usually, information retrieval systems are tuned for specific applications; thus, supporting either precision- or recall-oriented retrieval. Next, we describe how variable-context similarity assessment can be used to support flexible retrieval while ensuring high precision and high recall.

Variable-context similarity assessment assures that all cases that satisfy the current context are similar with respect to the context. Thus, precision and recall is 100% with respect to the given context. However, we have to take into account that the original query may be relaxed or restricted. Thus, we make a distinction between precision according to the query ($precision_q$) and precision according to the user ($precision_u$). Similarly, we distinguish recall with respect to a query ($recall_q$) from recall according to a user's original request ($recall_u$).

After making this terminological distinction, we return to the high precision and recall claim. We can prove that $precision_q$ and $recall_q$ is 100%. However, $precision_u$ and $recall_u$ may or may not be equal to 100%, depending on the context.

Theorem 6.1.1 *Context relaxation causes $precision_u$ to decrease (or remain the same), while con-*

text restriction causes precision$_u$ to increase (or remain the same).

Proof:

The proof follows directly from the monotonicity theorem Theorem 3.6.2.

□

This has an obvious advantage for the retrieval function, particularly during iterative browsing. The explain function uses a similar principle for controlled knowledge mining.

Accuracy and Coverage

Because precision and recall assess relevance of retrieved items subjectively they cannot be used for a direct comparison between different systems. However, we may evaluate the quality of a retrieval system by using accuracy, which is a common measure in machine learning.

Accuracy measures the quality of a classification system, i.e., the probability that a system correctly classifies a random item. Given a problem description the system classifies an item into a proper category, or predicts a value for its attributes. Some systems use training examples to create decision rules to be used during classification. Case-based classification uses past examples directly at the time of classification. This requires retrieving similar past cases and adapting them to fit the current problem's description.

Case-based classification assumes that similar cases have similar classification. So if the system retrieves strictly relevant cases then classification based on this knowledge would be more accurate. Thus, accuracy also measures the quality of the retrieval function and the capability of the adaptation algorithm. The retrieval function ensures high precision and high recall while the adaptation algorithm ensures the proper answer from the given set of relevant cases. If a system classifies with 100% accuracy, it is able to retrieve only relevant cases (assuming that the problem is not trivial).

The competence of a case-based classification system can be tested by removing test examples from a case base and trying to determine their attribute values (for example, classification for a case). Thus, we have an objective relevancy measure – a deviation between the actual suggested value can be expressed. This method is called *leave-one-out*, meaning that the system does not use the particular case during the reasoning process.

Although accuracy addresses the issues of recall and precision – dependency on subjective relevance measures – there are still obstacles which prevent us from direct comparison of accuracy results from one system and one domain to other systems, working in different domains. This is caused by the fact that accuracy is system and domain dependent:

1. **System dependency:** A better system achieves better accuracy. A system achieves a higher competence when only and all relevant knowledge is retrieved, compared to a system with low precision and/or recall.

2. **Domain dependency:** Some domains are inherently complex and this affects system competence and scalability. If the domain is not complete or there are missing attribute values in the case description, then even a well-designed system may have poor competence.

Due to domain dependency, there is a need to compare individual systems on the same problem domain or to devise measures that determine the complexity of the particular domain. Kononenko and Bratko suggested a fair evaluation criterion to measure classification accuracy (Kononenko and Bratko, 1991). The study was motivated by the fact that a simple comparison of classification accuracy might be misleading. Assume a domain with two classes, where 95% of all cases belong to one class. Then, a simple majority rule would have 95% accuracy with 5% misclassifications.

Several domain characteristics affect classification accuracy (Kononenko and Bratko, 1991): number of instances and target classes, number and types of attributes per instance, correlation of attributes to target concept disjuncts, distribution of instances among concepts, amount and type of noise present, and informativeness of the domain.

A perfect classifier achieves 100% accuracy. According to measures presented by Kononenko and Bratko, a perfect classifier in a 2-class domain is taken as a reference point (Kononenko and Bratko, 1991). With an increased number of classes in a domain, the percentage of correct classifications might drop. Using these measures, a classifier is perfect if in a domain with more than twenty classes it achieves at least 25% accuracy.

In such domains it is necessary that the system "knows" its boundaries and will not produce an incorrect answer. This issue strongly depends on the application domain and on system architecture. If the application is mission critical, then producing a wrong answer is usually much worse than not producing an answer at all. If the system is a stand-alone application, then not producing an answer means giving no help to the user. However, if the system works as a component of an underlying reasoner, (e.g., robot path planner) then not producing an answer might just delay the final solution.

High accuracy is easier to achieve when the system covers depth rather than breadth of a particular problem domain. We can select most appropriate problem by an "80-20 rule".[1] Coverage measures the portion of the problem domain where the highest accuracy is maintained.

6.1.2 Issues to be Addressed

In order to evaluate system performance one must decide what data sets to use and what measures to consider (Aha, 1992b; Buchanan, 1995). In the previous section we discussed precision, recall and accuracy performance measures. Here we describe some key issues:

[1] 80-20 rule refers to the fact that usually, 20% of problems occur 80% of the time (or cause 80% delay/cost). Thus, when designing a decision support system, it is best to address a small set of problems that have the biggest impact.

- **Relevance.** There are many definitions of relevance. What is relevant for one user (or in one situation) might not be considered relevant for another user (or in another situation).
- **Goal of retrieval.** Users might prefer precision- or recall-oriented retrieval.
- **Domain dependency.** Both system competence and scalability depends on the characteristics of the domain (number of instances and types of attributes per instance, distribution of instances, etc.)

Relevance

In different domains and for different systems there are distinct definitions of relevance (Greiner, 1994). Moreover, what is relevant in one situation, or for one user, may or may not be relevant in another situation, or for another user, (Jurisica, 1997). To handle both issues, the retrieval system should support dynamic definition of relevance. Assuming that a relevant item is a useful item, variable-context similarity-based retrieval helps achieve the needed flexibility. However, assessing the relevance may not be straightforward because one could measure it with respect to:

- user's belief – a domain expert (or a panel of experts) will decide which items are relevant, given a particular information base, task, and a query;
- initial query – regardless of the user's beliefs, the query is assumed to define relevance;
- automatically relaxed or restricted query – if the initial query did not retrieve the required quality and/or quantity of items, then it is transformed, which changes the relevance.

Measuring relevance with respect to a user's belief is difficult, since the user's intentions might not be expressible by the available query language or in a schema used. Thus, two repository items might be believed to be distinct, yet their representation could be the same. This makes retrieval difficult because both items would be recalled at the same time but one of them could be irrelevant.

Relevance can be measured directly with respect to the query. If the query is not relaxed or restricted, the system retrieves only (and all) relevant items. Note that if retrieval is iterative (query-by-reformulation) and the representation schema is sufficiently rich, the initial query should converge to the user's belief.

Measuring relevance with respect to a transformed query is needed during iterative browsing. Generally, automatic relaxation decreases precision, while user-guided relaxation may not.

One way to overcome the subjectivity of relevance measures is to use simple cross-validation (the leave-one-out method).[2] Although double cross-validation[3] is a better estimate of actual system

[2]This methods specifies that we test the procedure on data different from those used to choose its numerical coefficients (Mosteller and Tukey, 1977). Thus, in a CBR system, a case is selected at random to be a query, and its attribute values are predicted based on cases from a case base.

[3]According to this method the procedure on data different both from those used to guide the choice of its form and those used to choose its numerical coefficients (Mosteller and Tukey, 1977).

performance on real problems, simple cross-validation is appealing, since relevance is defined. Moreover, if the retrieval component is a pre-processing module of a reasoning system (e.g., classifier, diagnostic system) then prediction accuracy can be used to measure retrieval process quality.

Goal of Retrieval

The goal of retrieval might change for different tasks (Gravano, García-Molina and Tomasic, 1994). Retrieval can be either *precision-* or *recall-oriented.* In precision-oriented retrieval, the emphasis is put on retrieving only but not necessarily all relevant items, provided that no irrelevant items are retrieved. Recall-oriented retrieval emphasizes recalling all relevant items, i.e., irrelevant items are accepted as long as no relevant items are missed.

Traditional information retrieval systems support either recall- or precision-oriented retrieval. Thus, we trade off high recall for low precision and vice versa. Flexible retrieval systems should support both precision- and recall-oriented retrieval, since different users have different preferences, and these preferences may change over time while solving diverse tasks. The $\mathcal{TA}3$ system achieves flexibility by supporting two relaxation strategies – reduction and generalization (see Section 5.3.2). Consecutive context relaxations lead to iterative browsing, which increases precision while preserve high recall (Chu et al., 1996; Gaasterland, Godfrey and Minker, 1991; Jurisica and Glasgow, 1997).

Domain Dependency

From our experience with several problem domains (Jurisica and Glasgow, 1997) we have determined that system performance depends on the size and structure of the domain (both competence and efficiency is affected). On the one hand, if the schema for the domain is rich (i.e, it has sufficient informativeness), then individual items are separable and the performance results show the system's actual capabilities. On the other hand, for low informativeness, retrieval is negatively affected.

This raises an issue of how to compare individual systems. Unless they are evaluated on the same domain and using the same queries, the comparison is not an exact measure of their respective competence. Usually, sharing a real repository by various parties is not possible. This problem may be diminished by studying the structure of a real-world repository and then generating an artificial repository with equivalent characteristics (Aha, 1992b). This makes performance evaluation more controllable and enables the evaluation of several systems under the same conditions. An alternative is to use a domain independent measure of classification accuracy (Kononenko and Bratko, 1991).

6.1.3 Hypothesis and Scenarios

Our hypothesis is that the proposed system can be used as a flexible decision support system for the following tasks:[4]

1. **Classification:** using variable-context similarity assessment and an attribute-value pairs representation of cases diminishes the effect of irrelevant attributes on system performance – competence and efficiency. Thus, higher accuracy can be achieved.

2. **Prediction:** similarity-based retrieval can be used to access relevant past experiences, which in turn can be used to suggest attribute values for a case.

3. **Knowledge discovery:** variable-context similarity assessment helps to organize domain knowledge by grouping relevant attributes together and identifying representative values for these attributes.

4. **Case base organization:** variable-context similarity assessment can help identify equivalence classes of cases; depending on a given context, different sets of cases would be considered relevant.

5. **Information retrieval:** variable-context similarity assessment can improve retrieval from a (software) repository by recalling relevant information even if the user cannot specify the query completely and precisely. In addition, incremental context transformation supports iterative browsing and helps the exploration of the neighborhood of a given repository item.

Following sections evaluate the competence of $\mathcal{TA}3$. In the next chapter we evaluate $\mathcal{TA}3$'s efficiency. We show how it can be applied to different domains and support decision making in diverse tasks: (1) prediction in medicine – suggesting a cost-effective treatment for *in vitro* fertilization (IVF) patients without compromising the treatment outcome (Jurisica and Shapiro, 1995; Jurisica et al., 1998); (2) learning control – solving the inverse kinematics task for a three-link spherical, angular robot (Jurisica and Glasgow, 1996b; Jurisica and Glasgow, 1997); (3) classification into a continuous class – predicting the rise time of a servo-mechanism in terms of two (continuous) gain settings and two (discrete) choices of mechanical linkages (Jurisica and Glasgow, 1996b; Quinlan, 1993); (4) iris flower classification; (5) letter recognition (Frey and Slate, 1991; Jurisica and Glasgow, 1997); and (6) software reuse (Daudjee and Toptsis, 1994; Jurisica, 1997) – similarity-based retrieval of software artifacts.

We compare quality of solutions of the $\mathcal{TA}3$ system to other learning algorithms that have been applied to the same domains for the same task. In addition, we evaluate $\mathcal{TA}3$'s competence

[4] Additional details on applications areas for the $\mathcal{TA}3$ system are presented in Appendix A.6.

characteristics by using various adaptation techniques and by varying the relaxation process. For such an evaluation we pose the following hypotheses:

1. Generalization yields more accurate classification than reduction.
2. Reduction using attributes grouped only to one category yields less accurate classification compared to reduction when attributes are sensibly grouped into several categories.
3. Genetic algorithms can improve the system's competence by enlarging small case bases.

6.2 Case Studies

Here we discuss performance evaluation of $\mathcal{TA}3$ on diverse domains - medicine, robotics, botany, pattern recognition and software engineering. We also evaluate $\mathcal{TA}3$'s applicability to various tasks - prediction, learning control, classification, recognition, knowledge mining and information retrieval.

6.2.1 Prediction and Knowledge Mining in IVF

Here, we evaluate performance of the retrieval and explain functions on the IVF domain (described in Chapter 4) in terms of the quality of results the system can achieve.

Evaluation of the Retrieval Function

To evaluate prediction accuracy statistically, we conducted a series of tests, similar to the examples presented above. In order to get objective results, we used "leave-one-out" testing and conducted a series of random evaluations with 95% confidence intervals. The performance evaluation process was carried out by repeating the following steps: (1) select a random case and use it as a query-by-example; (2) retrieve relevant cases and order them according to similarity and the pregnancy outcome using pregnant patients as positive examples and other patients as negative examples; (3) compute an average value for the hormonal therapy from positive examples.

The results presented in Table 6.1 cover the first stage of prediction, i.e., predicting the hormonal therapy. Prediction accuracy was averaged over 20 random trials with a 95% confidence level.

Predicted Attribute	*Average Abs. Error*	*Relative Error*
NO_HMG	3.2 ± 1.98	0.15 ± 0.09
DAY_HCG	0.9 ± 0.558	0.07 ± 0.04

Table 6.1: *Accuracy of predicting hormonal therapy. Average and absolute errors between suggested and actual values for the treatment, namely day of human chorionic gonadotrophin administration (DAY_HCG) and the number of ampoules of human menopausal gonadotrophin (NO_HMG). Statistical results with 95% confidence level.*

During the process of predicting the pregnancy outcome, the system achieved an accuracy of 60.6%, averaged over 20 random trials. The accuracy of predicting pregnancy outcome was increased to 71.2% when 15 additional attributes were used, namely a series of estrogen (*E2*) values. This result is important to note. First, it suggests that the knowledge about IVF (and other complex domains) evolves over time. Second, it shows that as more information becomes available (i.e., as the informativeness of the domain increases), system competence also increases.

Evaluation of the Explain Function

Knowledge mining in $\mathcal{TA}3_{IVF}$ is user-directed, i.e., the user specifies the context and $recall_u$, which constrain the search space. $\mathcal{TA}3_{IVF}$ uses context reduction to organize cases for flexible retrieval, which in turn improves prediction accuracy and scalability. Moreover, new relationships among attributes may be discovered, and the case base can be structured into clusters of relevant cases.

In $\mathcal{TA}3_{IVF}$, the user poses a focused query, for example "What do most pregnant women have in common?", and the system finds a context which makes a given class of cases similar. Thus, $\mathcal{TA}3_{IVF}$ finds salient attributes and values for these attributes, i.e., a generalized interpretation of a given set of cases. This process is user-directed, interactive and possibly iterative. The user specifies the query, which constrains the search space, and the threshold T, which represents the certainty factor. The initial query is evaluated to produce a subset of the case base (e.g., find cases where $PREG : Y$ and $ABORTION : N$). Then, cases within the selected group are analyzed to produce a case interpretation that guarantees coverage T (e.g., at least T% of cases have protocol $PROT1 : 1, 2$). The result is confirmed by an alternative query (e.g., for cases where $PREG : Y$ and $ABORTION : Y$ determine $PROT1 : 1, 2$).

Table 6.2 shows partial results of knowledge mining on the set of patients that become pregnant and did not abort. The first column displays the initial context, selected from the first case in the set. The second column shows a modified context that satisfies the threshold $T \geq 50\%$. Attribute values that achieve high coverage with minimum constraints are likely to be good predictors for pregnancy. Attributes with lower coverage, i.e., high variance in their constraints, are not good predictors of pregnancy. As illustrated, some value constraints were relaxed to define a stronger pattern[5] (i.e., *DIAGNOSIS1*), some remained unchanged (i.e., *PROT*), and some were changed, since the initial value did not satisfy any of the selected cases (i.e., *PROT1*).

The coverage gives us the level of uncertainty for a particular attribute. It should be noted that the algorithm also confirms the result by evaluating alternative queries, such as pregnant patients with abortion. For the result to be a good predictor, this is necessary – specifying *PROT1: 0* for all pregnant women without abortion could also mean that in the cases matched all patients had *PROT1: 0*, even those without abortion. Using an alternative query showed that pregnant women

[5] A stronger pattern is an attribute constraint that satisfies the coverage threshold and has minimal variance.

Attribute Name	*Initial Context*	*Context Discovered*	*Change*
DIAGNOSIS	82%: vendo, tubal	51%: tubal	stronger pattern
DIAGNOSIS1	79%: 1-4	62%: 1	stronger pattern
DIAGNOSIS2	72%: 0	72%: 0	no change
NO_CYCLE	37%: 1	79%: 1-2	coverage increase
PROT	89%: LF	89%: LF	no change
PROT1	0%: 1	100%: 0	new constraint
BCP	65%: 20-30	65%: 20-30	no change
NO_HMG	62%: 14-18, 24-37	62%: 14-18, 24-37	no change
DAY_HCG	13%: 14-15	55%: 12-13	new constraint
ENDO_HCG	20%: 8-10	65%: 9-11	shifted constraint
CYC_OPU	12%: 16-17	55%: 14-15	new constraint

Table 6.2: *The context discovery function. Initial and final contexts for the class of pregnant patients. Initial context may remain unchanged if values for a given attribute satisfy the coverage constraint. New constraint may be created if old values are completely off. A constraint may be shifted to increase the coverage, or the constraint may be shrank to make it stronger.*

with abortion had *PROT1: 1* in 66% of the cases. Similar results have been obtained during a physician's interaction with the system.

The knowledge discovered was used to: (1) organize attributes into categories, (2) specify the context, and (3) generalize or specialize attribute value constraints. Thus,the classification accuracy obtained by $TA3_{IVF}$ serves as a measure of the knowledge discovery ability.

6.2.2 Inverse Kinematics Task

Here, the task is to compute the angles of the joints on a robotic arm so that the end-effector coordinates will reach a desired position. This task is computationally intractable for complex systems. The complexity increases with the number of robot arms: with just two arms and no additional constraints, there are at least two solutions for the inverse kinematics task. Hence, other methods are used to approximate the solution or to constrain the task. Approximation methods include fuzzy and neural net control (Tzafestas, 1995; Balakrishnan and Weil, 1996). The search space can also be constrained by limiting the energy consumption, maximizing the speed or the accuracy. Due to the practical nature of these problems, data availability and objective relevance measure, similar tasks have been used to test machine-learning algorithms (Aha and Salzberg, 1994).

We used an existing robot as an example. The inverse kinematics task covers a non-linear phenomenon – predicting the joint angles for the three-link spherical, angular robot when given desired end-effector coordinates. A spherical, angular robot is a universal robot with a large working area (see Figure 6.1). The robot's parameters are presented in Table 6.3, where L_1, L_2, L_3 are the lengths of the links, $\varphi_1, \varphi_2, \varphi_3$ are the angles between these links, which may have different ranges.

The equations that compute the end-effector coordinates P (forward kinematics task) are:

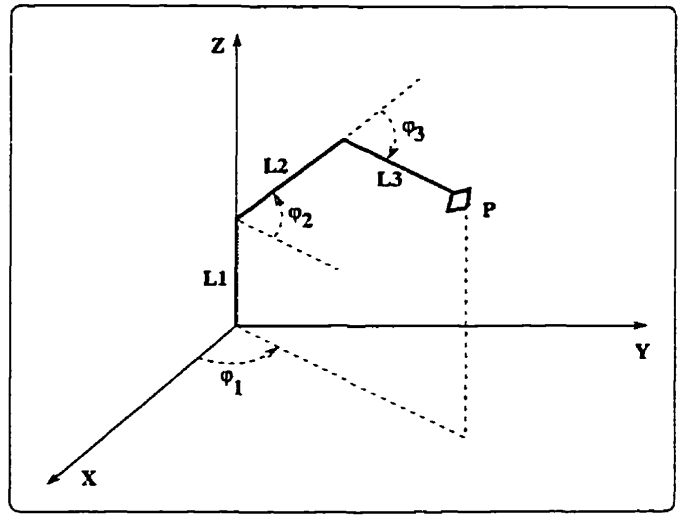


Figure 6.1: *Spheric angular robot.* L_1, L_2, L_3 *are the lengths of the links,* $\varphi_1, \varphi_2, \varphi_3$ *are the angles between these links, which may have different ranges.*

L_1	L_2	L_3	φ_1	φ_2	φ_3
0.203 m	0.178 m	1.178 m	$(-90°, 90°)$	$(-35°, 145°)$	$(-150°, 0°)$

Table 6.3: *Robot characteristics.*

$$
\begin{aligned}
P_x &= L_2 \cos\varphi_1 \cos\varphi_2 + L_3 \cos\varphi_1 \cos(\varphi_2 + \varphi_3) \\
P_y &= L_2 \cos\varphi_1 \cos\varphi_2 + L_3 \sin\varphi_1 \cos(\varphi_2 + \varphi_3) \\
P_z &= L_1 + L_2 \sin\varphi_2 + L_3 \sin(\varphi_2 + \varphi_3)
\end{aligned}
$$

The inverse kinematics computation may not yield only one solution. Moreover, complex kinematics structures may not have an analytical solution. This would require some optimality criteria to be defined. The other possibility is to use previous knowledge and compute only increments from the current position. Nevertheless, the equations for inverse kinematics of a given robot are:

$$
\begin{aligned}
\varphi_1 &= \arctan\frac{P_y}{P_x} \\
\varphi_2 &= \arctan\frac{P_z - L_1}{\sqrt{P_x^2 + P_y^2}} \pm \arccos\frac{L_2^2 - L_3^2 + P_x^2 + P_y^2 + (P_z^2 - L_1)^2}{2L_2\sqrt{P_x^2 + P_y^2 + (P_z^2 - L_1)^2}} \\
\varphi_3 &= \pm\left[\pi - \arccos\frac{L_2^2 + L_3^2 - P_x^2 - P_y^2 - (P_z^2 - L_1)^2}{L_2 L_3}\right]
\end{aligned}
$$

The obvious need for the inverse kinematics task is during planning - the robot's end-effector is

at point X and needs to be moved to point Y. For example in welding, the usual task is to follow a particular curve during the process. Thus, the inverse kinematics equations must be computed reasonably fast. We propose that by storing solutions to the inverse kinematics task and using CBR, a reasonable competence can be achieved. Furthermore, we can accommodate acquisition of new experience and thus improve system competence over time.

Case Base

Since we have the robot parameters and the necessary equations, we create several databases using different criteria (varying database size, distribution of instances, etc.). The case base for solving the inverse kinematics task can be characterized as follows:

- 2,000 cases;
- 9 attributes per case:

Attribute Name	*Description*	*Type*
P_x	x coordinate of the end-effector	float
P_y	y coordinate of the end-effector	float
P_z	z coordinate of the end-effector	float
L_1	length of the first link	float
L_2	length of the second link	float
L_3	length of the third link	float
φ_1	base angle	float
φ_2	shoulder angle	float
φ_3	elbow angle	float
X, Y, Z	coordinate frame	float

In the first experiment the whole case base was created uniformly, i.e., all 2,000 cases were uniformly distributed over the working space of the robot. This is useful for off-line learning to guarantee a reasonable answer in any possible situation.

The other possibility is to generate cases randomly. This is not useful in general because in a real-world environment there are no random moves of the robotic arm.

The last possibility is to use learning-by-observation, i.e., to produce cases during normal operation of the robot (Kuniyoshi, Inaba and Inoue, 1994). The problem is that this approach requires on-line operation. Furthermore, the working space may not be represented evenly. Thus, the system can operate only in situations similar to the ones observed during learning. The competence degrades when the task is changed. On-line learning may expand the case base generated off-line.

Evaluation of the Retrieval Function

During performance evaluation we varied several parameters, such as size of the case base, density and distribution of individual cases. In addition, we used various adaptation techniques and applied different relaxation processes.

Experiments were conducted using different criteria to select relevant cases. In general, case representation was organized into three Telos-style categories, one of which contained the class. An example of a context is shown in Table 6.4. During evaluation we used simple cross-validation (leave-one-out method) and we computed significance intervals and variances from the average values over 10 random trials (see Table 6.5).

Category	*Attribute Name*	*Attribute Value Constraint*	*Relaxation*
1	P_x	-0.118510	reduction
1	P_y	0.031099	reduction
1	P_z	0.217709	reduction
2	L_1	0.203000	reduction
2	L_2	0.178000	reduction
2	L_3	0.178000	reduction
3	φ_1	D_{φ_1}	
3	φ_2	D_{φ_2}	
3	φ_3	D_{φ_3}	

Table 6.4: *The inverse kinematics task - An example of a specific context.*

Method	*Average Absolute Error*			*Relative Error*		
	φ_1	φ_2	φ_3	φ_1	φ_2	φ_3
$\mathcal{TA}3_{Robot\ 1}$	0.4855 ± 0.2128	0.0	0.0	0.0059 ± 0.0025	0.0	0.0
$\mathcal{TA}3_{Robot\ 2}$	0.0	2.5 ± 0.0120	0.0	0.0	0.0275 ± 0.012	0.0

Table 6.5: *The inverse kinematics task - absolute and relative errors on robotic data using simple cross-validation; average over 20 random trials (95% confidence intervals).*

Only reduction was used during context relaxation in $\mathcal{TA}3_{Robot\ 1}$ (i.e., for the first two categories, only m-of-3, $1 \leq m \leq 3$, attributes were required to match). $\mathcal{TA}3_{Robot\ 2}$ used generalization of attribute value constraints first, and if this failed, then reduction was performed. Thus, attribute values were relaxed using numerical proximity, where increments for generalization were chosen based on other attribute values for the same attribute. It is interesting to note that $\mathcal{TA}3_{Robot\ 1}$ outperformed $\mathcal{TA}3_{Robot\ 2}$, although the former uses a domain-independent relaxation method.

6.2.3 Learning Control

We used a freely available learning control data set,[6] which has been used as a test set for various machine-learning algorithms (Quinlan, 1992). Servo data addresses a non-linear problem: predicting the rise time of a servomechanism in terms of two (continuous) gain settings and two (discrete) choices of mechanical linkages. Thus, the rise time is the class with infinite number of values.

Case Base

The servo data set can be characterized as follows:

- 167 cases;
- 5 attributes per case:

Attribute Name	*Attribute Values*
motor	A,B,C,D,E
screw	A,B,C,D,E
pgain	3,4,5,6
vgain	1,2,3,4,5
class	real number from 0.13 to 7.10

All attributes in the case are placed into three categories: (1) - *motor* and *screw*; (2) - *pgain* and *vgain*; and (3) - *rise time*. Table 6.6 shows a context example used during case retrieval.

Category	*Attribute Name*	*Attribute Value Constraint*	*Relaxation*
1	motor	{A, B}	reduction
1	screw	{C, D}	reduction
2	pgain	4	reduction
2	vgain	2	reduction
3	rise time	$D_{rise\ time}$	

Table 6.6: *Servo mechanism domain - Context example.*

Evaluation of the Retrieval Function

Although this is not a complex or large domain, it is of interest to us, since various machine-learning algorithms have been evaluated and compared with respect to it. Table 6.7 summarizes results for the $\mathcal{TA}3_{Servo}$ and other machine-learning techniques, obtained by using simple cross-validation (leave-one-out method) and averaged over 10, 40 and 60 random trials respectively.

[6] A description of the data set, and references to results obtained with other machine-learning algorithms is available at ftp:ics.uci.edu:pub/machine-learning-programs/servo/.

Method	*Average Abs. Error*	*Relative Error*
Guessing mean	1.15	1.00
Instance-based	0.52	0.26
Regression	0.86	0.49
Model trees	0.45	0.29
Neural nets	0.30	0.11
Regression+instances	0.48	0.20
Model trees+instances	0.30	0.17
NN+instances	0.29	0.11
$\mathcal{TA}3_{Servo\,1}$ (60 trials)	0.118	0.073
$\mathcal{TA}3_{Servo\,1}$ (40 best trials)	0.111	0.085
$\mathcal{TA}3_{Servo\,2}$ (60 trials)	0.09	0.056
$\mathcal{TA}3_{Servo\,2}$ (40 best trials)	0.009	0.006

Table 6.7: *Servo-mechanism domain - Absolute and relative errors on servo-data using simple cross-validation; average over 20 random trials.*

$\mathcal{TA}3_{Servo\,1}$ uses generalization where only the *motor* attribute is relaxed and only to the left side (e.g., B is relaxed to A *or* B). $\mathcal{TA}3_{Servo\,2}$ applies generalization only on the *motor* attribute, but in both directions (e.g., B is relaxed to A *or* B *or* C). It should be noted that relaxing A and E in both scenarios is equivalent – relaxing upwards. In situations where the initial relaxation does not yield a result, we can apply reduction on an attribute category that includes *motor* and *screw*. We can then apply reduction on attribute category with *pgain* and *vgain*.

Method	*Average Abs. Error*	*Relative Error*
Instance-Based	9.29 times	4.6 times
NN + Instance-Based	3.2 times	1.96 times

Table 6.8: *Improvement of $\mathcal{TA}3_{Servo\,2}$, 6 10-WCV over neural nets + instance-based and plain instance-based approaches.*

It is not sufficient in general to determine performance of different systems by direct error comparison. Table 6.7 lacks statistical evidence – there is no information on either the confidence in the results, the significance of the errors, or the confidence intervals. Thus, we conducted a statistical evaluation of our system, which is presented in Table 6.9.

Tables 6.8 and 6.9 show that it is advantageous to be able to change the relaxation strategy, and also that the servo domain has better informativeness than the IVF domain. Moreover, we can see that not even 40 trials is enough for 99% confidence with a reasonably small confidence interval.

Method	Average Abs. Error		Relative Error	
	99% Confidence	95% Confidence	99% Confidence	95% Confidence
$TA3_{Servo\,1}$ (60)	0.118 ± 0.042	0.118 ± 0.032	0.073 ± 0.032	0.073 ± 0.024
$TA3_{Servo\,1}$ (40)	0.111 ± 0.051	0.111 ± 0.038	0.085 ± 0.043	0.085 ± 0.033
$TA3_{Servo\,2}$ (60)	0.09 ± 0.035	0.09 ± 0.027	0.056 ± 0.012	0.056 ± 0.009
$TA3_{Servo\,2}$ (40)	0.009 ± 0.008	0.008 ± 0.006	0.006 ± 0.004	0.006 ± 0.003

Table 6.9: *Simple cross-validation - results averaged over 40 and 60 random trials of* $TA3_{Servo}$.

6.2.4 Iris Flower Classification

The iris flower data set is publicly available on a machine-learning database server.[7] The data set was used to test various machine-learning algorithms, mainly for classification and feature selection (Cheeseman et al., 1988; Wettschereck, Aha and Mohri, 1995; Ricci and Avesani, 1995).

Case Base

The iris data set contains three classes of 50 instances each, where each class refers to a type of iris plant. Only one class is linearly separable from the other two. Given four numeric attributes, the task is to predict the class of iris plant. The data set can be characterized as follows:

- 150 instances;
- 5 attributes per instance;
- class distribution is 33.3% for each of the three classes.

Attribute Name	Attribute Values
sepal length	4.3-7.9cm
sepal width	2.0-4.4 cm
petal length	1.0-6.9 cm
petal width	0.1-2.5 cm
iris plant	Iris Setosa, Iris Versicolour, Iris Virginica

Evaluation of the Retrieval Function

We tested the retrieval function by measuring classification accuracy. Given the four attributes, the system predicted the class of the iris plant. We focused on testing the competence of the system, and how it is influenced by the the following factors: (1) attribute category membership; (2) relaxation strategy; (3) case base size; and (4) genetic algorithms.

[7] A description of the data set and references to results obtained with other machine-learning algorithms are available at ftp:ics.uci.edu:pub/machine-learning-programs/iris/.

In order to demonstrate the capability of the system, we grouped attributes into three categories: class, sepal and petal measures. We also used a representation, where all attributes but class were put into the same category. In addition, we tested several relaxation scenarios:

- First use generalization; if no classification is possible, apply reduction as well.
- Use reduction only – we tested two modifications: (1) with three categories per instance; and (2) with two categories per instance.

We performed tests with various case base sizes: first using a complete case base, then a case base with randomly removed cases. We also tested how a genetic algorithm affects competence, when a cross-over function enlarges a case base. In this experiment (referred to as Iris_{30GA34}), the cross-over point was chosen by separating the first two attributes from the rest of the case description.

Table 6.10 summarizes performance on the basis of ten plant descriptions selected at random. It should be noted that all incorrect or undecided classifications occurred for the Iris Virginica plant. Some of the classifications required several iterative relaxations; first, category 0 was relaxed, then category 1, and finally both categories were relaxed. During generalization, additional reduction was required only once. The system used only simple generalization – immediate neighbors of an attribute value were used as interval borders, i.e., 1.2 would be relaxed to a range from 1.1 to 1.3.

Iris_{150}			
Generalization	*Reduction*		*Classification*
	3 Categories	*2 Categories*	
100%	80%	90%	Correct
0%	0%	10%	Incorrect
0%	20%	0%	Undecided
Iris_{60}			
100%	60%	80%	Correct
0%	0%	20%	Incorrect
0%	40%	0%	Undecided
Iris_{30}			
100%	40%	80%	Correct
0%	0%	20%	Incorrect
0%	60%	0%	Undecided
Iris_{30GA34}			
90%	50%	90%	Correct
0%	0%	10%	Incorrect
10%	50%	0%	Undecided

Table 6.10: *Iris domain for various case base sizes – classification accuracy on predicting the class of iris plant, using simple cross-validation; average over 10 random trials (95% confidence intervals).*

As hypothesized earlier, generalization achieves 100% accuracy on all case base sizes, except when the genetic algorithm was used to enlarge the case base. Even then, however, the system

did not produce an incorrect classification, instead it returned an "Undecided" answer. Reduction yields less accurate classification when attributes are grouped only to one category, compared to reduction when attributes are sensibly grouped into several categories. Although accuracy is lower when all attributes are grouped into three categories, the important distinction is that the system does not produce incorrect classification as it does when only two categories are used. This feature is especially useful for critical - e.g., medical - applications. There is a slight improvement in competence when the case base size is increased by a genetic algorithm (see the comparison of Iris_{30} and Iris_{30GA34}).

A hybrid reasoning system that combines CBR and genetic algorithms was tested on this domain (Skalak, 1993). The author reports above 95% accuracy with less than 5% of all cases, by using only prototypical cases. Fertig and Gelertner (1991) report 76.6% accuracy on the same data . Ricci and Avesani (1995) report accuracy of 92.8%. They also report that the initial case base size can be reduced using an asymmetric anisotropic similarity metric algorithm, while preserving the same accuracy. It should be noted that using $\mathcal{TA}3$'s similarity-based retrieval reduces the number of cases in the case base five times, while preserving 100% accuracy with generalization.

Evaluation of the Explain Function

We evaluated the explain function on the feature selection problem, using the same data set. The task is to reduce the number of features that describe data, while not decreasing the competence significantly (Aha and Bankert, 1995; Wettschereck, Aha and Mohri, 1995).

The explain function selected petal measures as important features for classification. Sepal measures were not linearly separable and thus could not be used alone for classification. It should be noted that the system did not create class characteristics for 100% coverage, since the second and third iris plant class cannot be separated. Instead, the border was found to be in between. If only one petal measure attribute is used for classification, then some Versicolour plants would be classified as Virginica and vice versa. This problem can be diminished in one of the two ways:

1. If the border values are separated completely, the accuracy would be increased. However, the system would not be able to recognize the border cases.

2. If both petal measure attributes are used during classification, accuracy is increased and the coverage is not decreased.

By choosing either method, the system can be fine-tuned to be recall- or precision-oriented. The final decision on system configuration depends on the task for which classification is performed. If on the one hand, the system is supposed to give an informative classification, then the recall-oriented approach would be more suitable. Thus, classes can be overlapped, provided that coverage improves.

On the other hand, if the task is critical, then a precision-oriented approach is preferred. In such situations, classes should be separated completely, even at the price of decreased coverage.

6.2.5 Character Recognition

The character recognition domain is publicly available data set.[8] The data set was used to test various machine-learning algorithms (Frey and Slate, 1991; Aha, 1992b). The task is to identify each of a large number of black-and-white rectangular pixel displays as one of the 26 capital letters of the English alphabet. The character images are based on 20 different fonts. Each letter within these 20 fonts was randomly distorted to produce a file of 20,000 unique stimuli (Frey and Slate, 1991). Each stimulus was converted to 16 primitive numerical attributes which were then scaled to fit into a range of integer values from 0 through 15. Most of the systems that used this data set were trained on the first 16,000 items. Accuracy is measured by applying the learned model to the remaining 4,000 letters for letter category prediction (Frey and Slate, 1991; Aha, 1992b).

Case Base

The character recognition data set can be characterized as follows:

- 20,000 instances;
- 16 attributes per instance (type of attributes: integer 1..16):

[8] A description of the data set and references to results obtained with other machine-learning algorithms are available at ftp:ics.uci.edu:pub/machine-learning-programs/letter.

Attribute Name	*Attribute Domain*	*Attribute Values*
Class: letter	26 letters	A to Z
horizontal position of a box	integer	1 – 15
vertical position of a box	integer	1 – 15
width of a box	integer	1 – 15
height of a box	integer	1 – 15
total number of pixels	integer	1 – 15
mean x of on pixels in a box	integer	1 – 15
mean y of on pixels in a box	integer	1 – 15
mean x variance	integer	1 – 15
mean y variance	integer	1 – 15
mean x y correlation	integer	1 – 15
mean of x * x * y	integer	1 – 15
mean of x * y * y	integer	1 – 15
mean edge count left to right	integer	1 – 15
correlation of x-edge with y	integer	1 – 15
mean edge count bottom to top	integer	1 – 15
correlation of y-edge with x	integer	1 – 15

- number of target classes: 26;
- number of prototypes per class: 20;
- distribution of instances among concepts: uniform;
- distribution of instances among prototypes: normal.

Evaluation of the Retrieval Function

For the classification task the system is presented with 16 attributes that describe a letter. Given a case base of previously seen cases the task is to identify the letter. The results presented in Table 6.11 were obtained using a simple cross-validation (leave-one-out) method and averaged over twenty random trials.

In Table 6.11 all algorithms but $\mathcal{TA}3_{Letter}$ were trained on the first 16,000 and 1,600 cases respectively and then tested on the last 4,000 cases. Reported results are average values over ten trials. Since $\mathcal{TA}3_{Letter}$ does not require a learning stage, we used leave-one-out testing on the first 2,000 and on all 20,000 cases. Results are averaged over twenty random trials.

In $\mathcal{TA}3_{Letter\,1}$ we used a modified nearest-neighbor approach – attributes were grouped into categories for selective reduction. $\mathcal{TA}3_{Letter2}$ was obtained by using generalization first (i.e., relaxing

Method	*Accuracy (20,000 cases)*	*Accuracy (2,000 cases)*
Back-propagation	—	81.9 ± 0.6%
IB1	95.7 ± 0.4%	81.7 ± 0.7%
CN2	87.9 ± 0.8%	68.7 ± 1.0%
C4	86.4 ± 0.7%	67.4 ± 0.8%
Genetic Algorithm	82.7	—
$\mathcal{TA}3_{Letter\,1}$ over 20 trials	90% ± 0.1461	80% ± 0.2011
$\mathcal{TA}3_{Letter\,2}$ over 20 trials	100%	85% ± 0.1742
$\mathcal{TA}3_{Letter\,3}$ over 20 trials	100%	95% ± 0.1001
$\mathcal{TA}3_{Letter\,4}$ over 20 trials	100%	100%

Table 6.11: *The character recognition domain - absolute and relative errors on servo-data using simple cross-validation; average over 20 random trials (95% confidence intervals).*

the attribute values to include values of immediate neighbors). If more cases were needed, we applied reduction. $\mathcal{TA}3_{Letter\,3}$ was obtained by using a combination of previous approaches with equal voting for each of them, i.e., the final answer was composed of the two results obtained by the respective context relaxation approaches (reduction and generalization), with equal voting for each of them. $\mathcal{TA}3_{Letter\,4}$ was obtained by using different category groupings for attributes. In the previous test, categories were created sequentially, each group having four attributes (the letter attribute was an extra category). In the latter test we used an explain function to find the most relevant attributes to recognize individual letters and we grouped attributes accordingly.

6.2.6 Retrieval from a Software Repository

Software repositories are used to store and retrieve information about software artifacts. Designing a software repository requires that three issues be considered: (1) information content of a repository; (2) information representation; and (3) strategies for accessing repository artifacts efficiently. Here we concentrate on the last issue. The repository must support different tasks and diverse users efficiently. Thus, retrieval strategies cannot be pre-defined in the system and the system should support imprecise queries and exploratory browsing of the repository content.

This case study is different from previous ones, because it addresses the user issue more explicitly. For this purpose, we include some background information on problems considered, and some user studies that define requirements for the retrieval system. Next, we show how variable-context similarity assessment can be efficiently used to support various users in solving diverse tasks. In addition, to further enhance handling of imprecision during the retrieval process we extend variable-context similarity assessment by defining lexical similarity.

We have identified tasks pertinent to novice and expert software engineers within a specific project. A novice user's main task is familiarization. The software repository should assist novices in fast and in-depth learning. A hierarchical access to repository artifacts and imprecise queries

must be supported.

Expert users have domain knowledge, and their retrieval strategies are task-dependent. Experts are involved in one of the following tasks:

1. *Re-documentation and design recovery* – the user wants to understand a system, design decisions taken and the logic behind the implementation.

2. *System migration* – the user wants to translate a system from one language to another, change software architecture, or port a system to another platform.

3. *Maintenance and evolution* – the user needs to improve system functionality and fix existing problems.

4. *System testing* – the user wants to analyze system correctness (i.e., competence) and efficiency.

5. *Requirements re-engineering* – the user needs to extract, analyze, understand and modify system requirements.

Generally, software artifacts can be accessed by selecting them in diagrams, by specifying a query (a search-based approach), by specifying a sample artifact (a query-by-example approach), and by modifying the earlier query (a query-by-reformulation approach). These approaches are further extended to include similarity-based retrieval.

Traditionally, information retrieval has been concerned with representation of texts and comparison of these representations. Software engineering research contemplated abstracting source code to make the retrieval process easier by providing extra information (Canfora, Cimitile and Munro, 1996; Faustle, Fugini and Damiani, 1996; Termsinsuwan, Cheng and Shiratori, 1996). In addition, users rarely specify queries precisely and/or completely. As a result, uncertain and imprecise information must be taken into account when designing retrieval systems. Numerous approaches support imprecise queries, most of which are based on fuzzy logic (Ruspini, 1991; Vila et al., 1996).

It is expected that a software repository supports diverse information retrieval problems. Thus, different interaction methods and diverse information-seeking strategies must be available. Retrieval strategies may change not only between individual sessions but even within a particular one. Users rarely specify their information needs, mainly because conception of information problems may change in the process. More recently it was observed that information retrieval is an inherently interactive process (Belkin et al., 1995; Jurisica, 1994). Thus, the retrieval system should deploy an efficient iterative browsing (Carpineto and Romano, 1996; Maarek, Berry and Kaiser, 1991), which provides the means for exploring the "neighborhood" of a repository artifact. Such neighborhood depends on the task, current state of the search, and the repository state. This means that the neighborhood of an artifact is context-dependent.

Next, we show how variable-context similarity assessment can be used to handle imprecise queries, and to find information relevant to a result set of a given query. In addition, query-by-reformulation is supported by automatic relaxation and restriction algorithms.

For example, `dsinit` is a function – a repository artifact with the following attributes:

```
name:              dsinit,
formal arguments:  (int, float),
return type:       float,
source location:   dsini.
```

These attributes specify a function signature. Since not all attributes are equally important for retrieval, they should be placed into different categories. The semantics of these attributes suggest three categories: {`name`}, {`formal arguments`, `return type`} and {`source location`}. Hence, the semantic information of attribute value types is defined through the use of context.

A context specifies important (or relevant) attributes and how "close" an attribute value must be in order to satisfy the context. Given the `dsinit` function, one can specify a context as follows:

```
name:              {dsinit},
formal arguments:  {int, float},
return type:       {float, int},
source location:   {dsini}.
```

Using the same set of attributes, one can specify a context, which is satisfiable by any function that returns `float` and is part of the `dsini`:

```
return type:       {float},
source location:   {dsini}.
```

Since `name`, `formal arguments` and `return type` are not part of the context, they are ignored during matching. Technically, we assume that those attributes have a set of allowed values as their domains:

$$\begin{array}{ll} \texttt{name:} & \{D_{name}\}, \\ \texttt{formal arguments:} & \{D_{arguments}\}, \\ \texttt{return type:} & \{\texttt{float}\}, \\ \texttt{source location:} & \{\texttt{dsini}\}. \end{array}$$

Recall that we say that an object satisfies (or matches) a particular context (a query) if for each attribute specified in the context the value of that attribute in the object satisfies the constraint (i.e., the value is contained in the constraint set). In the proposed theory, similarity is considered as a relation between objects with respect to a specific context.

A component retrieval scenario can be defined as a query specification, followed by linear retrieval and browsing (Maarek, Berry and Kaiser, 1991) . The usefulness of different information-seeking

strategies depends on the user's background and her task. Available strategies can be characterized on many dimensions, each with varied combination of characteristics (Belkin et al., 1995):

Method of interaction:	Scanning	-	Searching
Goal of interaction:	Learning	-	Selecting
Mode of retrieval:	Recognition	-	Specification
Resource considered:	Information	-	Meta-Information

Assuming different user models and active tasks, it is possible to estimate a user's goal, and then suggest an appropriate information-seeking strategy (Jurisica, 1997). Individual strategies may be interchanged even within one session. For example, a novice user is more likely to get better results using a scanning method of interaction. However, after retrieving the relevant information, then she may use searching method or query-by-example to further refine the information or re-focus the search. In contrast, a maintenance software engineer needs an efficient searching method in order to retrieve a specific repository artifact. This in turn can lead to using recognition mode of retrieval instead of specification. Also, the goal of interaction for a novice user is learning, while for an expert user it is selecting. Sixteen information-seeking strategies can be defined using the dimensions presented. Each of these strategies fits a particular user model. For example, a user searching a repository directory with a goal of selecting artifacts relevant to a particular task is characterized by the scanning method of interaction, using the recognition mode of retrieval, and searching through meta-information. A system-knowledgeable user can use meta-information to retrieve specific artifacts. This is achieved when method of interaction is searching, the mode of retrieval is specification, and the goal of interaction is selecting.

When designing the retrieval facility, we aimed at supporting various information-seeking strategies, so users with different tasks would be able to use the system effectively. The context for similarity-based retrieval can be defined by the user, inferred from the user's current actions, or transformed (relaxed or restricted) by the system. When context is defined by the system, a particular user model can be used to provide information about the user's primary task, her preferences, search patterns, previous tasks, etc. Recently accessed repository artifacts provide contextual information such as location (spatial similarity), data structures used, and data-flow dependencies.

Variable-context similarity assessment supports iterative browsing by incremental query relaxations and restrictions (Jurisica and Glasgow, 1998). Browsing is crucial in software library systems (Maarek, Berry and Kaiser, 1991), and is useful in training usage or when a repository is large and created by many users over a longer period of time. This allows for designing a cooperative retrieval system, similar to one described in (Chu et al., 1996).

Similarity-based retrieval deploys concept hierarchies of artifacts, and uses virtual links to traverse the repository and to relax or restrict the query. For example, when accessing a variable declaration, the system retrieves all modules and files that include such a declaration. In addition,

similarity-based retrieval locates files and modules where this variable is fetched or stored. It also accesses variables that use the same data type. Although the system can perform query transformations automatically, the process is interactive and cooperative; the user controls the query transformation via changes to the context.

To achieve both high recall and high precision the proposed variable-context similarity assessment uses iterative browsing and features provided by Telos. ore specifically, categories group equally important attributes together. Thus, each attribute has its own constraint and can be treated in a distinct way during relaxation and restriction. This results in a more flexible version of the m-of-n matching (Ortega, 1995), which enables higher precision (Jurisica and Glasgow, 1997). Incremental query revisions are computed efficiently (Jurisica and Glasgow, 1998).

Heuristics can also be used to improve precision and recall. User studies revealed specific patterns that are followed by experts during searching for information (Jurisica, 1997). For example, experts usually search for store operations first and then for fetches.

There are several levels of query imprecision. In addition, different representation schemes can be used to store repository artifacts. To cover the spectrum of these problems, we define the following measures used in the similarity-based retrieval module: lexical, semantic, functional, and spatial.

Lexical Similarity. Lexical similarity includes textual similarity, in particular name matching. There are two possible approaches: a contrast model (Tversky, 1977) and N-gram indexing. For the contrast model, assume $\mid P_{\mathcal{C}} \mid$ represents the length of the query pattern and $\mid P_i \mid$ represents the length of the repository artifact pattern. We define $\mid P_{\mathcal{C}} \wedge P_i \mid$ as the length of the pattern that is present in both $P_{\mathcal{C}}$ and P_i. Then the lexical similarity is defined as follows:

$$sim(P_{\mathcal{C}}, P_i) = \frac{\mid P_{\mathcal{C}} \wedge P_i \mid}{\mid P_{\mathcal{C}} \mid + \mid P_i \mid + \mid P_{\mathcal{C}} \wedge P_i \mid}.$$

Lexical similarity handles only lexical problems. The contrast model or n-gram indexing does not find xx_string and xx_text to be similar. Given a query xx_init and files: xx_intg, xx_dsini, xx_string and xx_point from one package, the contrast model selects xx_intg and xx_point as the closet matches. A simple 3-gram selects xx_intg as the closest match, followed by xx_dsini and xx_point. When assuming naming conventions, a simple heuristic can be used to select xx_dsinit as the closet match to xx_init. This selection is coherent with the semantic similarity of respective files.

Semantic Similarity. The problem of lexical similarity that arises with the example of xx_string and xx_text can be diminished by using background knowledge and *part-of* relationship. Since string is part of text, xx_string and xx_text are similar with respect to *part-of* relationship. This, however, requires semantically rich lexicon. Currently, this kind of matching is implemented by having the user state such relations in context.

Semantic similarity also enables us to find variables that use similar data structures as returned value or a function parameter. Distance between artifacts is determined using *is-a* and *part-of* hierarchies of data types. This approach is similar to (Termsinsuwan, Cheng and Shiratori, 1996) but variable-context similarity assessment controls, which features of the data type description are more important for a specific search. This distinction is crucial for both high recall and high precision. Next we describe how semantic and functional similarities are related.

Functional Similarity. Functional similarity assesses relevance of artifacts on the basis of their functional descriptions (Faustle, Fugini and Damiani, 1996). Our system can use functional similarity once the abstracted description is provided. The similarity is determined between a functional description in the query and functional descriptions found in the repository, taking into account relevance of individual features. The comparison is achieved using pattern matching on function signatures (Jurisica, 1997). For example, the following signature specifies functions of any name, integer return type, parameters of type integer, float and bit value, and that can be in any module:

⟨ `name:` D_{name},
`formal arguments:` `(int, float, bit value)`,
`return type:` `int`,
`source location:` $D_{location}$⟩.

The matching process can be tuned up by using information about relevance of features. Lexical similarity is used as a flexible approach to wildcards matching (Faustle, Fugini and Damiani, 1996). It controls the matching process semantically, and supports incremental query transformations.

Spatial Similarity. Spatial similarity is determined based on the spatial relevance, i.e., placement in the same module, file or function, and is determined using a knowledge base (Finnigan et al., 1997). *Part-of* relationship can be used to assess semantic similarity. For example `flow.ss` (flow subsystem) is semantically similar to `support.ss`, because both subsystems are *part-of* the `dead_store` module. Similarly, the `memmap`, `memdsc`, `plshadow` and `refoccs` data structures are semantically similar, because all are used in the `dead_store` module (i.e., they are *part-of* `ds.pl9`).

Case Studies

Here we describe how the variable-context similarity assessment extends traditional retrieval techniques. Text-based search is enhanced by supporting lexical similarity, and by the ability to search in a neighborhood of a repository artifact. Faceted search is improved by supporting query transformations, and iterative browsing. Relevance of the retrieved artifact is determined with respect to the user task, preferences, and the current state of the search. Next we discuss several tasks in which a software engineer may be involved. We show how variable-context similarity assessment supports

the process. The studies presented are based on interviews of novice and expert users working on a specific project.

System Familiarization. Novice users do not have enough domain knowledge to support problem-oriented search through the repository. Thus, the method of interaction is scanning and the mode of retrieval is recognition. They need to acquire knowledge about basic data structures used in the project, and form an overall system design representation. As the users become more familiar with the studied system they acquire meta-information in the repository. The main objective for the retrieval system is to decrease the overall time needed for familiarization.

Since the novice user's goal is to understand a complex system they need to traverse the repository hierarchically. For these reasons, query-by-example, query-by-reformulation, and iterative browsing are suitable retrieval strategies for novices.

Finnigan et al. describe scenarios, where a novice navigates through the repository by following links from a system architecture diagram, obtains a subsystem overview, and retrieves subsystem details by selecting one of its possible views (Finnigan et al., 1997). Similarity-based retrieval enhances the current tools by supporting approximate queries, and browsing through conceptually neighboring software artifacts.

The variable-context similarity-based retrieval system can further assist the user by suggesting query selection, by performing query transformations (relaxation or restriction), and by providing additional relevant information. This helps to both control the amount of information presented to the user, and explore the neighborhood of a particular software artifact (using semantic or spatial similarity). Both issues are crucial in helping a novice create a mental model of the system faster.

Assume the following query:

```
name:              {dsinit},
formal arguments:  {int, float},
return type:       {float},
source location:   {dsini}.
```

If the user wants to explore other relevant artifacts, then the query must be relaxed. The following relaxation strategies are available:

- *Lexical similarity* may be used to find lexically similar artifacts.
- *Reduction* can be used to explicitly remove attributes from the context, e.g., eliminate source location from the matching process. Another possibility is to dynamically reduce the number of required attributes, i.e., to apply m-of-n matching.
- *Generalization* may be applied by using *is-a* and *part-of* hierarchies. Another possibility is to use expansion of the set of allowed values.

For example, relaxing the constraint on the `source location` attribute can be generalized from {`dsini`} to {`dead_store`}, using the following piece of domain information: `dsini` *part-of* `dead_store`. Another possibility is to expand the set of allowed values, e.g., we can change `return type:{float}` to `return type:{int, float}`.

Program Development. Developing a software system requires familiarity with its code. Since the source code for a large system is usually stored in a hierarchical tree structure, navigation through it may be difficult. It is not uncommon to "*spend one to three days looking for a certain definition, only to write one new line of code*" (Jurisica, 1997). It was estimated that in about "*three quarters of a year, one month is spend on searching for information*" (Jurisica, 1997).

Another problem with developing a large system is that many developers are involved. This means, that code changes are frequent. On a specific project it was observed that "*in about 2.5 hours, some 50 files were modified*" (Jurisica, 1997). Thus, a desirable retrieval mechanism should support imprecise query specification and similarity-based retrieval. This can help locate relevant artifacts without overloading a user with irrelevant information. Due to the frequency of code-change, running a cross-reference utility once a day might not provide sufficient information. However, similarity-based retrieval finds relevant matches using lexical similarity or generalization.

Generally, queries are direct, e.g., find a specific variable definition. However, there is also a need for imprecise queries to deal with the problem of change and lack of domain knowledge, and similarity-based retrieval to consider conceptually neighboring software artifacts.

Assume a developer looking for a function that is probably called `ds_mrgs`, since it is part of the `dead_store` module. Using lexical similarity the user finds that there is a function called `dsmrgs`. Functional similarity is an alternative way to locate `ds_mrgs` using its signature:

$$
\begin{array}{ll}
\langle\ \texttt{name:} & D_{name}, \\
\texttt{formal arguments:} & \texttt{(int, float, bit value)}, \\
\texttt{return type:} & \texttt{integer}, \\
\texttt{source location:} & D_{location}\rangle.
\end{array}
$$

Let us assume that the user found the specified function and thus `name:dsmrgs` and `source location:dsini` (the remaining parts of the signature remain unchanged). This forms an initial query-by-example. If the user needs to locate other similar functions, there are several options for query relaxation. First, reduction can be used: it is sufficient if either `name` or `source location` is matched, provided that both `formal arguments` and `return type` match. In addition, reduction can be applied to `formal arguments` – it would be sufficient if only two-out-of-three arguments match. If the user reduces the signature to `source location:dsini`, then in effect spatial similarity is used. Second, generalization can be used: `return type:integer` can be changed to `return type:number`. Alternatively, `formal arguments:(int, float, bit value)`

can be changed to `formal arguments:(number, number, bit value)`. Finally, `return type` can be changed to `return type:(integer, Boolean)`, meaning that either `integer` or `Boolean` type is considered a match.

Domain knowledge may be used to improve the query processing, by improving query transformations. For example, the tree structure of the source code is built with certain preferences. Also, another development team may use `.ch` files for in-line definitions. There is an interesting performance issue: if the `.ch` file is included in the header file (class definitions) then it results in a fast executable but a slower compilation; if it is included in a source file, then the compilation is faster but it results in slower executable. Other types of preferences come naturally from the task under consideration. For example, when a developer needs to find a variable, (s)he usually looks for a store operation first, then for a fetch. Thus, the system may predict this and pre-load information about the fetch operation right after retrieving a store operation.

Similarity matching can also be used in situations when a query is specified imprecisely. This may include situations, where the type of a variable is known but not its name. The retrieval tool needs to find the definition of all variables with a particular type, but can possibly order them based on their location in the tree. This may affect how relevant the variable from a particular module is in a given situation. This can be implemented via signature matching, as exemplified next.

Since many functions, class and variable definitions are defined using inheritance, an obvious question is how many levels should be traversed in order to answer the query. Because different tasks require different depth, the number of levels to be traversed should be selectable.

Migration/Reverse Engineering. Migration of the software system includes its porting to another computer architecture or to another language. During this process, an exact copy of the system in another language may be produced (transliteration process). Alternatively, a higher level migration, such as design migration and architecture migration can be done.

Transliteration is the process of directly translating a code written in one language to another. The process may be completely automated or it may be semi-automatic. This may result in inefficient code since different languages support various constructs with variable efficiency. A more difficult approach is to take data structures into consideration and to optimize them during system migration. More complicated issues arise when change of the system architecture system is required.

Migration and reverse engineering programmers face the problem of finding variable declarations efficiently and reliably. A recall-oriented retrieval process is needed, since the failure to retrieve all relevant artifacts may lead to incorrect decisions. Precision must also be high, to reduce information load on the user. Retrieval may be complicated by the fact that a function call might look identical to a variable (e.g., PL/I dialect), and the two may be distinguishable only from their declaration. Relevant queries include "Find a variable declaration", "Find a variable usage", and "Find a function

with a parameter of a specific data type".

Migration software engineers are supported using a two-phase approach. (1) All relevant information from the repository is retrieved. (2) The query is iteratively transformed, so irrelevant artifacts are removed. Another strategy is to find a relevant artifact via query-by-example and consider other related artifacts in the repository. Consider a signature of the following function:

```
⟨ name:             dsinit,
formal arguments:   (int, float),
return type:        float,
source location:    dsini ⟩.
```

Reduction with generalization can be applied to find other functions with both their arguments and return types being equal. Query reduction results in the following:

```
formal arguments:   {int, float},
return type:        {float}.
```

Generalization can change the first argument of the `formal arguments`from `int` to `number`:

```
formal arguments:   {number, float},
return type:        {float}.
```

Performance Evaluation

This section describes preliminary performance evaluation of the proposed similarity-based retrieval system. Our claim is that similarity-based retrieval improves precision, and supports iterative browsing. Our previous studies suggest that the claim holds (Jurisica and Glasgow, 1997; Jurisica et al., 1998). Here we consider a functional description repository of basic data structures (Daudjee and Toptsis, 1994; Faustle, Fugini and Damiani, 1996).

A functional description is a repository artifact, described by a set of features and organized into categories. Each feature is a triplet: ⟨ `verb, noun, weight` ⟩, where `verb` is the operation, `noun` is the object and `weight` indicates importance of the feature in the given functional description. A sample functional description with two features can be defined as follows:

```
(   ⟨ verb: insert, noun: node, weight: low ⟩,
    ⟨ verb: delete, noun: node, weight: high ⟩   ).
```

During retrieval, if no relevant artifacts are returned for the specified functional description, the query might be transformed, as described in Section 3. First, reduction is applied, followed by generalization. Generalization uses *part-of* and *is-a* hierarchies of terms, e.g., `node` *part-of* `b-tree`, `element` *part-of* `list`, `b-tree` *is-a* `tree`, etc. Such hierarchies make it possible to both specify imprecise queries and to explore a neighborhood of the currently retrieved artifacts. For example,

after retrieving a functional description, such as ⟨ verb:insert, noun:node, weight:low ⟩, one could explore its neighborhood, i.e., define a context that would retrieve other similar artifacts from the repository with respect to certain features. First, consider an operation to find other functional descriptions with the same operation:

verb: {insert},
noun: {element},
weight: {high}.

Second, find functional descriptions with a different operation on the object that is in addition similar to a given description:

verb: {delete},
noun: {node},
weight: {medium}.

During evaluation, a functional description was selected at random and submitted as in query-by-example (leave-one-method). We have evaluated recall and precision after averaging results over twenty random tests. Using explicitly stated context and query transformations we controlled the relevance (quality) and quantity of retrieved artifacts. By selecting reduction, generalization or both, we simulated recall- and precision-oriented retrieval, recall-oriented retrieval with iterative precision improvement. Using the last approach yielded 100% recall rate with perfect precision. These results are consistent with our previous evaluations (Jurisica and Glasgow, 1997; Jurisica et al., 1998), but since the repository of functional descriptions is rather small (only 150 data structures) and simplistic, we intend to run full-scale studies on a larger and more complex software repository.

Performance improvement over results presented earlier (Daudjee and Toptsis, 1994; Faustle, Fugini and Damiani, 1996) was achieved for two reasons. First, variable-context similarity assessment keeps semantic information about distance among artifacts, which enables better control over relevant and irrelevant artifacts. Second, a combination of the retrieval system with a high recall, and iterative browsing that removes irrelevant artifacts increases precision. Our approach differs from previous systems (Faustle, Fugini and Damiani, 1996; Maarek, Berry and Kaiser, 1991) because the semantic information is part of the similarity assessment process.

6.3 Discussion

Similarity-based retrieval tools can advantageously be used in building flexible retrieval and classification systems. Variable-context similarity assessment supports flexibility by: (1) letting the user to define different relevancy measures and retrieval strategies; (2) restricting or relaxing context automatically; and (3) supporting similarity-based queries, query-by-example and query-by-reformulation (Jurisica, 1996). Case-based classification uses previous instances to classify unknown

problems. Classification accuracy is affected by the retrieval process – the more relevant the instances used for classification, the higher the accuracy. Using different context and criteria affects the time spent on searching for the solution as well as the quality of the solution. $\mathcal{TA}$**3** allows for tuning up the reasoning to meet different requirements, similarly as in (Cunningham, Bonzano and Smyth, 1995).

Using the terminology of normative rationality introduced in (Good, 1952), the medical application of the $\mathcal{TA}$**3** system (Jurisica and Shapiro, 1995; Jurisica et al., 1998) follows type I rationality, i.e., inference that is consistent with the axioms of decision theory regardless of the cost of inference. A more accurate suggestion for the hormonal therapy has a positive impact on pregnancy and is also cost effective, since it minimizes the quantity of hormones given to the patient. Thus, even though the treatment should be suggested reasonably quickly, clearly accuracy is more important.

Robotic application of $\mathcal{TA}$**3** (Jurisica and Glasgow, 1996b) implements type II rationality: Since time resources are limited the cost of reasoning is considered. Thus, even an approximate solution provided in real time has a higher value than an accurate solution delivered late. In the inverse kinematics task, there are fast techniques available to produce an accurate solution from an approximate one. However, for some robotic architectures, computing inverse kinematics is intractable. Thus, the main objective is to produce a solution within time constraints.

A flexible system requires strategies that refine partial results continuously. $\mathcal{TA}$**3** supports this by context relaxation or restriction. Given partial results, the system changes m in m_of_n matching to increase or reduce the number of retrieved cases. By controlling this process, $\mathcal{TA}$**3** improves precision of retrieval, while preserves perfect recall.

Flexibility of the $\mathcal{TA}$**3** system is further enhanced by incorporation of the knowledge discovery algorithm that allows for finding optimal knowledge representation from two points of view: precision of results and efficiency of computation. Thus, importance weight of attributes may dynamically change, which in turn changes retrieval strategies.

$\mathcal{TA}$**3** is different from other CBR approaches, since it does not use predefined retrieval strategies. Instead, similarity assessment can be changed dynamically for a particular domain and specific application. Hence, $\mathcal{TA}$**3** can be configured to suit individual tasks. This enables flexible computation by trading off the accuracy or precision of the retrieval process for time resources. Advantages of $\mathcal{TA}$**3** can be summarized as follows:

1. Semantic similarity assessment during retrieval and explanation of inference process.
2. Case base organization that improves access time and makes visualization easier to understand.
3. User-oriented knowledge mining that may improve precision, recall and coverage.

Although the presented approach to similarity-based retrieval is part of the CBR system $\mathcal{TA}$**3**, the same principle is applicable to other systems that require information retrieval, and where items

are represented as attribute-value pairs. Thus, $\mathcal{TA}3$ can be used to build flexible retrieval systems.

The results presented support the claim that incremental context transformations result in higher competence of the reasoning system (Jurisica and Glasgow, 1998; Jurisica et al., 1998). In the next chapter we show that incremental context transformation is also efficient.

Chapter 7

Evaluating the Efficiency of $\mathcal{TA}3$

This chapter experimentally evaluates efficiency of $\mathcal{TA}3$ on application domains presented in Chapter 6, and scalability on the performance model of $\mathcal{TA}3$. We discuss efficiency of the retrieval algorithm, presented in Section 3. We compare scalability of standard and incremental implementation of the retrieval algorithm with respect to case base size, context complexity, case representation complexity, and number of iterations during iterative browsing.

7.1 Introduction

Various approaches can be used to evaluate system performance. Available methods evaluate either competence of the system[1] or its scalability. The former measures the capabilities of the system and can be assessed by precision/recall or accuracy/coverage measures. The latter one assesses scalability of the system based on the cost required for computing the result. System efficiency can be measured by directly comparing required time. However, such a measure is system dependent (Segre, Elkan and Russell, 1991). A better approach is to measure the cost required to perform individual operations, e.g., computer time required.

There are several factors affecting performance of a CBR system: the size of a case base, size of a case and context, query complexity, number of iterations during iterative browsing, and the query relaxation/restriction strategy used. In this chapter we present two efficiency evaluations of $\mathcal{TA}3$:

1. Experimental efficiency evaluation using application domains described in Chapter 6; and
2. Scalability evaluation using the performance model of $\mathcal{TA}3$.

[1] Competence evaluation of $\mathcal{TA}3$ is presented in Chapter 6.

Experimental Efficiency Evaluation of $\mathcal{TA}3$. During experimental evaluation of $\mathcal{TA}3$'s efficiency we considered average queries from application domains presented in Chapter 6. We varied the case base size, the context-transformation strategy and number of iterations during browsing. A computer time required for performing three consecutive context modifications was measured, using the same query for different case base sizes.

We tested competence of the implemented prototype system on several domains, details of which are presented in Chapter 6. Here we report on its efficiency and scalability, namely on the gains from incremental query modification, compared to naive handling of queries. The servo case base covers a non-linear phenomenon - predicting the rise time of a servomechanism in terms of two (continuous) gain settings and two (discrete) choices of mechanical linkages (see Section 6.2.3 for more details). The data set consists of 167 instances described by 5 attributes. We used $\mathcal{TA}3$ for the classification task. The accuracy we obtained was comparable to other machine-learning techniques. During relaxation only the *motor* attribute was relaxed but in both directions (e.g., B would be relaxed to A *or* B *or* C). In situations where the initial relaxation did not yield a result, additional methods can be used. First, the cardinality relaxation for *motor* and *screw* is tried. Second, the cardinality relaxation for *pgain* and *vgain* is tried.

The letter classification data set consists of 20,000 instances described by 17 attributes (see Section 6.2.5 for more details). Based on simple cross-validation (leave-one-out method), the $\mathcal{TA}3$ system achieved comparable classification accuracy to other systems. Attributes were grouped into categories for selective cardinality relaxation/restriction. We tested several relaxation strategies that yielded various accuracy results, with comparable cost.

Predicting the joint angles (φ_1, φ_2 and φ_3) for the three-link spherical angular robot when given desired end-effector coordinates (inverse kinematics task), is covered in the robotic domain (see Section 6.2.2 for more details). The presented results are for a uniformly generated case base, which consists of 2,000 instances.

The studied medical domain (*in vitro* fertilization) consists of medical records about patients (see Section 6.2.1 for more details). The case base, which is available to us, consists of 788 cases and 55 attributes per case (after confidential information has been removed).

Scalability Evaluation of $\mathcal{TA}3$. For scalability evaluation, we considered results of $\mathcal{TA}3$'s experimental efficiency evaluation on various real-world domains and complexity properties of retrieval algorithms (both naive and incremental) to create a performance model of $\mathcal{TA}3$, which is presented in the next section.

This model simulates $\mathcal{TA}3$'s scalability as a function of case base size, case representation complexity, query complexity, and context-modification strategy used. As a validation of this model, we compared experimental efficiency evaluation and scalability evaluation of $\mathcal{TA}3$ on respective case

base sizes, case representation complexity, and context complexity (see Figures 7.1 and 7.2).

7.2 The Performance Model of TA3

From the algorithms presented in Figures 3.7 and 5.3 it follows that k consecutive context modifications require k applications of the naive algorithm. Thus, a naive approach requires $k \times \mid \Delta \mid \times \mid \Omega \mid$ evaluations. In contrast to this, an incremental algorithm is evaluated only once to produce SI_{part} and subsequent context modifications are handled by incremental changes to the result.

Assuming that the initial retrieval result $\mid SI_{part} \mid$ is substantially smaller than the case base size $\mid \Delta \mid$, producing SI' incrementally is significantly more efficient. For all categories that are not affected by a context modification, no recomputation is necessary. For categories affected by a change, local changes handle the recomputation as presented in Section 5.4. It should be noted that both algorithms could be improved by indexing, which avoids accessing all cases in a case base.

Iterative retrieval is based on the algorithm defined in Figure 3.7. The complexity of iterative retrieval depends not only on the number of cases in a case base, context complexity, but on the user criteria, the number of retrieved cases, and the number of consecutive iterations. Next we present a performance model of individual context transformations, both for naive and incremental implementations.

Iterative retrieval with standard reduction depends on the case base size, context, and number of iterative context reductions:

$$\mid \Delta \mid (\mid \Omega \mid + k \mid \Omega \mid - \sum_{1}^{k} i),$$

where $\mid \Omega \mid$ is the context complexity, $\mid \Delta \mid$ is the case base size, and k is the number of iterations.

Iterative retrieval with incremental reduction depends on the case base size, context, set of returned cases, and number of iterative context reductions:

$$\mid \Omega \mid (\mid \Delta \mid + k \mid SI \mid - \mid SI \mid \sum_{1}^{k} i),$$

where $\mid \Omega \mid$ is the context complexity, $\mid \Delta \mid$ is the case base size, $\mid SI \mid$ is the number of returned cases, and k is the number of iterations.

Iterative retrieval with standard expansion depends on the case base size, context, and number of iterative context expansions:

$$\mid \Delta \mid (k \mid \Omega \mid + \sum_{1}^{k} i),$$

where $\mid \Omega \mid$ is the context complexity, $\mid \Delta \mid$ is the case base size, and k is the number of iterations.

Iterative retrieval with incremental expansion depends on the case base size, context, set of

returned cases, and number of iterative context expansions.

$$| \Omega | (| \Delta | + | SI | (k + \sum_{1}^{k} i)),$$

where $| \Omega |$ is the context complexity, $| \Delta |$ is the case base size, $| SI |$ is the number of returned cases, and k is number of iterations.

Iterative retrieval with standard generalization depends on the case base size, context, and number of iterative context generalizations:

$$| \Delta || \Omega | (k + 1),$$

where $| \Omega |$ is the context complexity, $| \Delta |$ is the case base size, and k is number of iterations.

Iterative retrieval using incremental generalization depends on the case base size, context, and number of iterative context generalizations:

$$| \Delta | (k + | \Omega |),$$

where $| \Omega |$ is the context complexity, $| \Delta |$ is the case base size, and k is number of iterations.

Iterative retrieval with standard specialization depends on the case base size, context, and number of iterative context specializations:

$$| \Delta || \Omega | (k + 1),$$

where $| \Omega |$ is the context complexity, $| \Delta |$ is the case base size, and k is number of iterations.

Iterative retrieval with incremental specialization depends on the case base size, context, set of returned cases, and number of iterative context specializations:

$$| \Omega | (| \Delta | + k | SI |),$$

where $| \Omega |$ is the context complexity, $| \Delta |$ is the case base size, $| SI |$ is the number of returned cases, and k is number of iterations.

Next we show efficiency and scalability evaluation using experimental results on presented application domains and the performance model of TA3. On the basis of experiments from real-world domains, we assumed that ten cases are returned in average, and that ten consecutive context transformations are performed. It should be noted that if fewer than ten cases are returned, or more than ten consecutive context transformations are performed, then the incremental approach is even more efficient than a standard one.

7.3 Efficiency and Scalability Evaluation of $\mathcal{TA}3$

Figure 7.1 shows how value/cardinality relaxation and restriction depends on the case base size. We compare both standard (S) and incremental (I) approaches. $\mid CB \mid$ represents case base size, G is a generalization and R represents reduction. Individual data points are created using the performance model of $\mathcal{TA}3$. These results are compared to the experimental efficiency evaluation of $\mathcal{TA}3$ on robotic, servo, medical and character recognition domains. (Figure 7.2 presents results only for the incremental approach.)

We have used the complexity of incremental implementation of context transformations (presented in Section 5.4.2) to create a model that fits $\mathcal{TA}3$'s performance on real-world domains, presented above. We parameterized this model to determine the scalability of $\mathcal{TA}3$ with respect to (1) case base size, (2) query complexity (context), and (3) number of consecutive iterations for individual context transformations. In the model we assumed that partial evaluations are kept on attribute level. Thus, more storage space is required, but the system is more efficient when frequent changes to the context are made.

Presented results support the claim that the incremental context manipulation is more efficient then the naive approach. Performance improvement is increased when the case base size is substantial, cases have many attributes and several subsequent context modifications are required. The results presented in Figures 7.3-7.6 show that the incremental approach improves performance by 70% on average. However, for simple retrievals - small case base and/or no consecutive context modifications - the naive approach would be more efficient.

7.4 Discussion

The results presented support the claim that incremental context transformations are more efficient than the naive approach (Jurisica and Glasgow, 1998). Performance improvement is increased when the case base size is substantial, cases have many attributes and several subsequent context transformations are required. However, for simple retrievals (case base and/or no consecutive context transformations) the naive approach would be more efficient, due to smaller memory requirements.

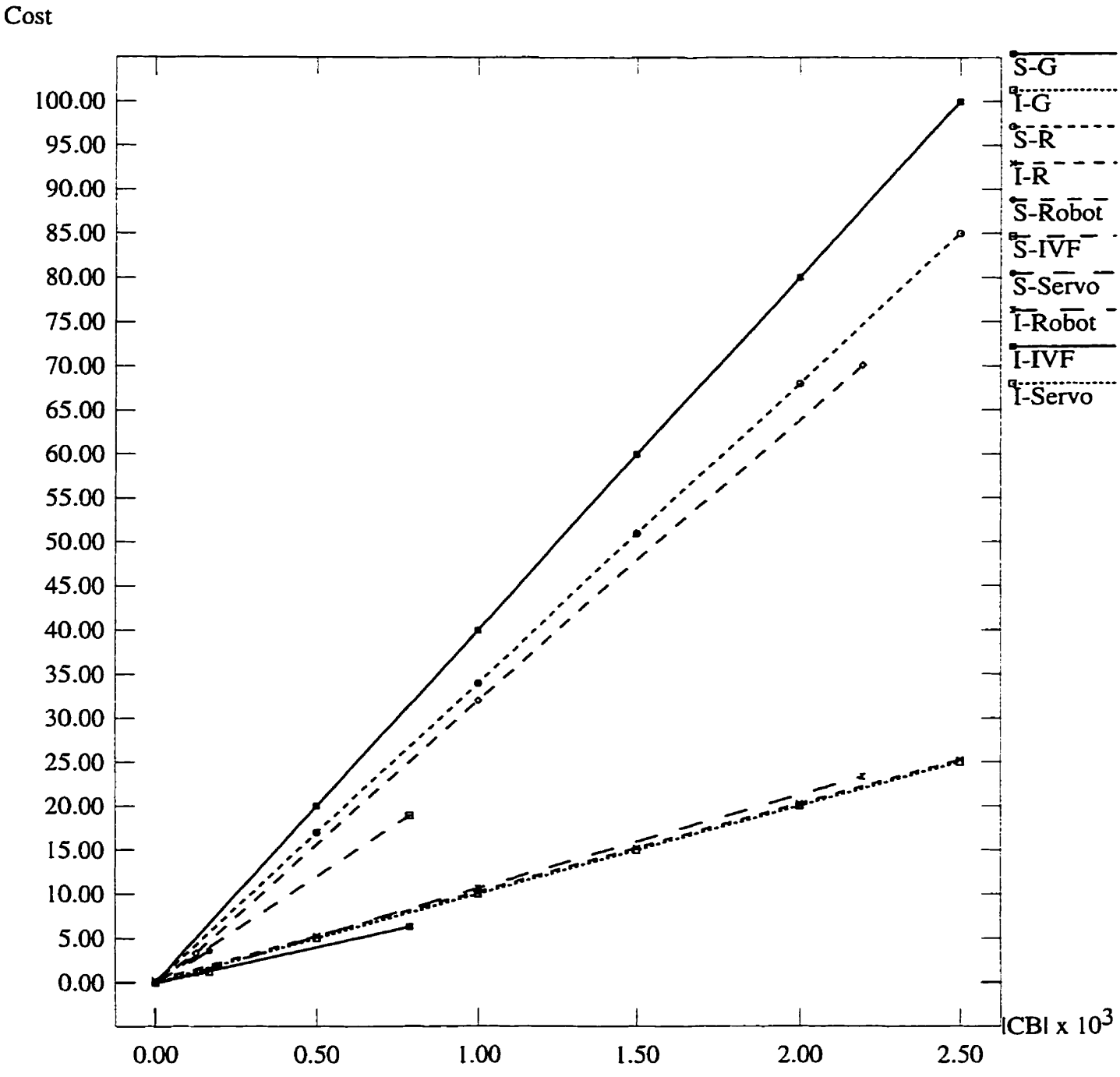


Figure 7.1: *Cost of retrieval for three context modifications. S/I - standard/incremental strategy; G/R - generalization and reduction.*

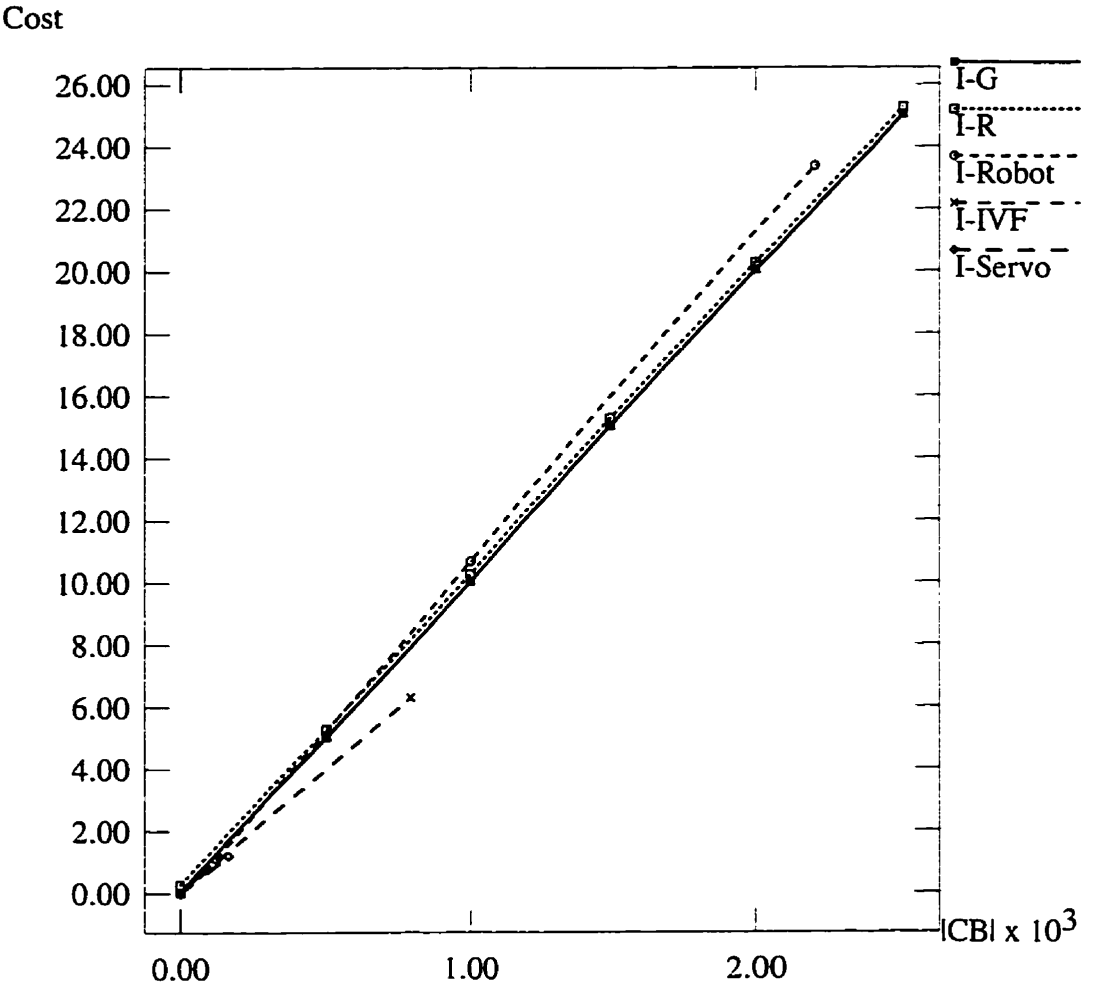


Figure 7.2: *Cost of retrieval for three context modifications. I - incremental strategy; G/R - generalization and reduction.*

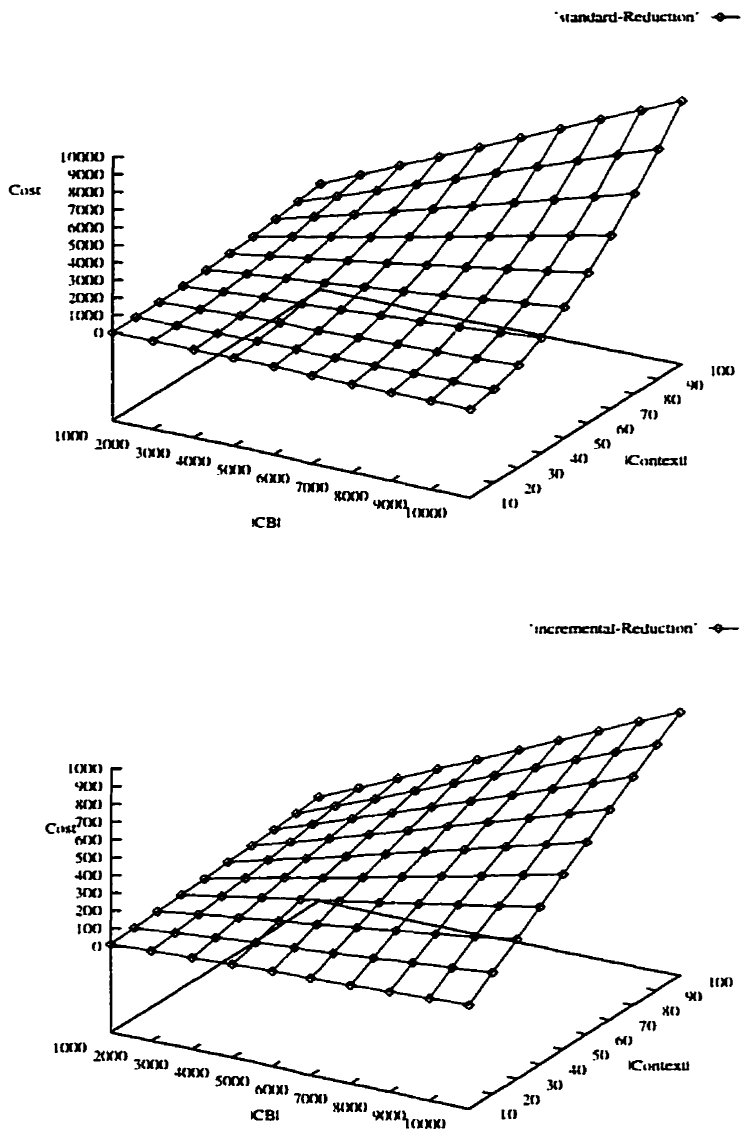


Figure 7.3: *Cost of retrieval for standard and incremental reduction as a function of case base size (| CB |) and size of the context (| Context |). Ten consecutive relaxations are considered.*

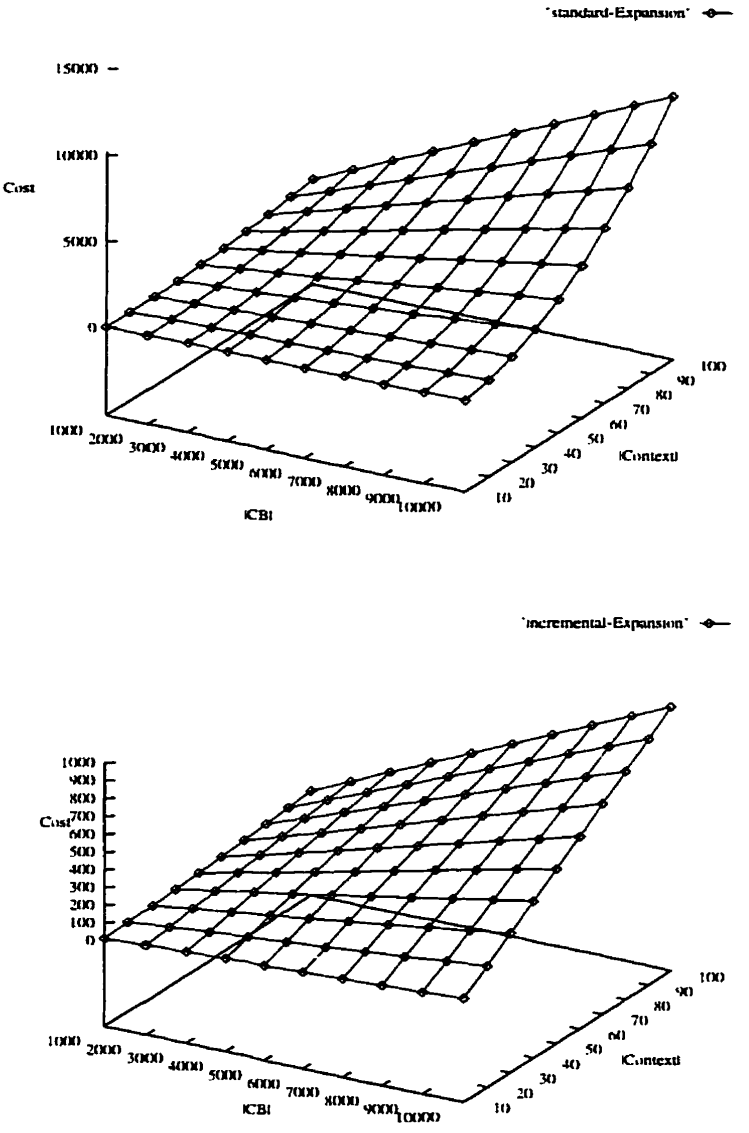


Figure 7.4: *Cost of retrieval for standard and incremental expansion as a function of case base size (| CB |) and size of the context (| Context |). Ten consecutive restrictions are considered.*

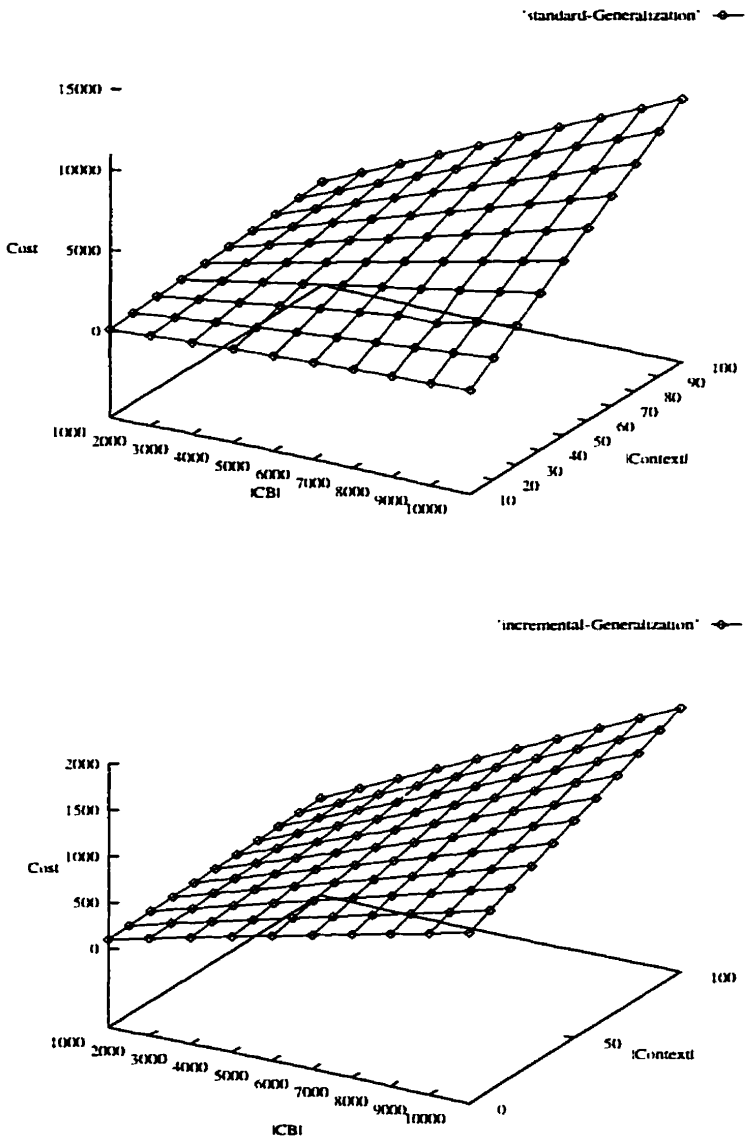


Figure 7.5: *Cost of retrieval for standard and incremental generalization as a function of case base size (| CB |) and size of the context (| Context |). Ten consecutive relaxations are considered.*

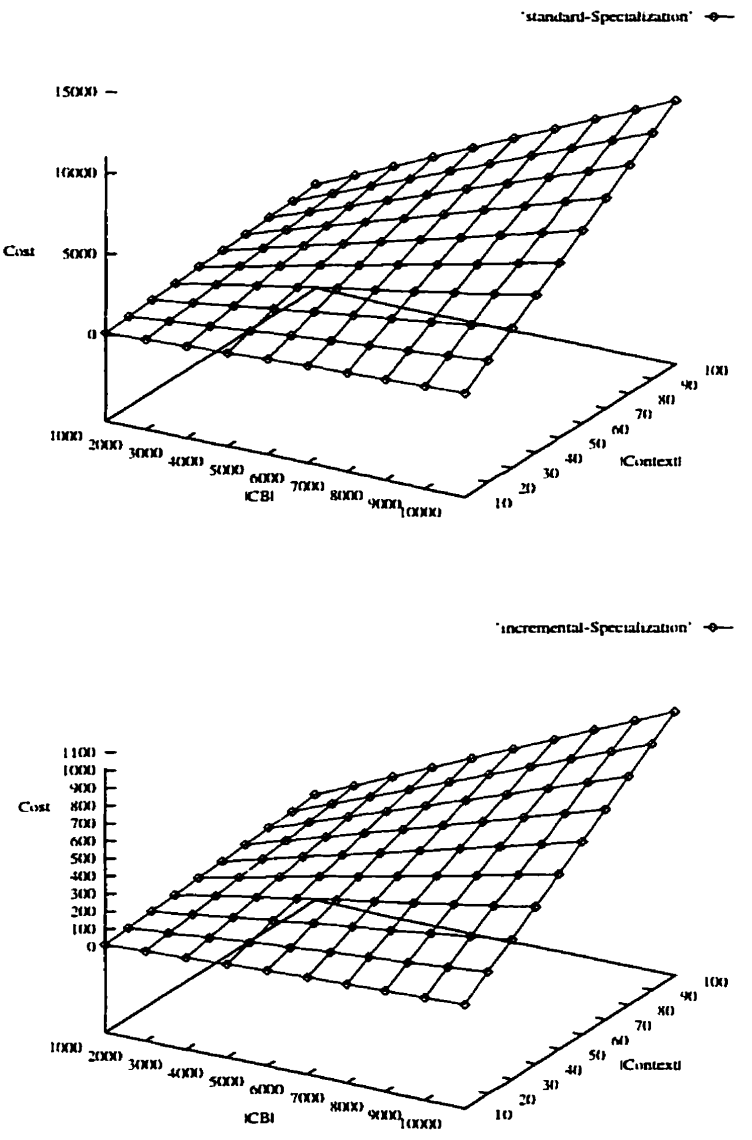


Figure 7.6: *Cost of retrieval for standard and incremental specialization as a function of case base size (| CB |) and size of the context (| Context |). Ten consecutive restrictions are considered.*

Chapter 8

Conclusions

8.1 Contributions of the Thesis

The contribution of the thesis is in taking a performance approach to CBR. We aimed at improving both quality of the results (system competence) and scalability of the system.

We show how variable-context similarity assessment (a) improves the quality of solutions in terms of higher classification accuracy; (b) extends system flexibility with respect to different tasks and various user models; and (c) supports scalable implementation of retrieval algorithms.

These contributions involve: (1) developing a new theory for similarity assessment using explicitly defined context; (2) studying properties of the proposed variable-context similarity assessment approach; (3) designing an efficient case retrieval algorithm using variable-context similarity assessment, and adopting view maintenance algorithms from database management systems for supporting iterative browsing; (4) evaluating the algorithmic complexity of proposed algorithms and their scalability with respect to case base size, complexity of case representation and query complexity; and (5) evaluating the performance of the proposed case-based reasoning prototype on diverse real-world domains, namely evaluating the competence of the reasoning system, its applicability to different problem solving tasks, and system efficiency with respect to application domain, task, case base size, case and query complexity.

The categorization of similarity assessment theories helped to identify strengths and weaknesses of individual approaches. On the basis of these results, we have proposed a similarity assessment theory, which subsumes and unifies previous approaches. In addition, variable-context similarity assessment improves system competence by using a modified nearest-neighbor retrieval and instance-based learning. This approach requires the use of a flexible case representation language. In turn, such a representation enables an efficient iterative browsing using incremental query modifications. These techniques are applicable beyond the CBR paradigm, as long as information is represented as

attribute-value pairs. Namely, query-by-example, query-by-reformulation, and iterative information retrieval with high recall and precision can all be supported.

Due to the flexible case representation used, the efficient case base organization and the efficient retrieval algorithm deployed, the proposed CBR system supports reasoning in multiple contexts. The underlying architecture enables an efficient implementation that permits scalable CBR, in terms of case base size, complexity of case representation and query complexity. These are some of the desired features of future AI systems (Hayes-Roth, 1997).

Efficient iterative browsing is supported via an incremental context transformation algorithm that draws on database techniques for incremental view maintenance. The results presented support the claim that incremental context manipulation is generally more efficient than the naive approach. It is effective to use an incremental context manipulation approach when a large case base is used, when cases have many attributes, or when several subsequent context modifications are required. In other words, this approach is most suitable for iterative browsing in complex domains. For simple retrievals involving a small case base and/or no consecutive context modifications, a naive approach may be preferable, since scalability is not an issue and no extra storage space is required.

The proposed incremental algorithm is general in the sense that it is applicable to retrieval in any decision support system, where information is represented as attribute-value pairs, and problems are solved by iteratively accessing and using previously derived information. Adopting machine-learning algorithms (namely attribute-based knowledge mining, reinforcement learning and forgetting) for improving the quality of results and speed of reasoning extends the core prototype system.

The quality of the proposed CBR system was evaluated on the basis of its competence, efficiency, and algorithmic complexity on real-world domains. We discussed suitable evaluation criteria for CBR systems, and compared quality of results obtained by using different versions of the proposed prototype to those obtained by other algorithms on the same domains. During evaluation we varied domain parameters (e.g., case base size, case and query complexity) and system parameters (e.g., relaxation strategy, type of matching). System performance was evaluated on domains with different characteristics (e.g., solution quality, efficiency, scalability), and the system was used to support diverse tasks.

The evaluation of the proposed system shows a potential to apply variable-context similarity assessment as a basis for diverse applications: query-less and query-by-example retrieval, iterative browsing, query-by-image-content, knowledge base organization, knowledge discovery, diagnosis, design, classification and prediction.

Using different contexts and criteria affects the time spent on searching for a solution as well as the quality of the solution. $\mathcal{TA}3$ can be tuned to meet different requirements, as suggested by Cunningham et al. (Cunningham, Bonzano and Smyth, 1995). On the one hand, in the medical domain (Jurisica and Shapiro, 1995; Jurisica et al., 1998), a more accurate suggestion for hormonal

therapy positively influences rates of IVF outcome, and reduces the risk of complications associated with the treatment. In addition, an accurate suggestion for hormonal therapy is also cost effective, since it minimizes the quantity of hormones given to the patient. Thus, even though the treatment should be suggested reasonably fast, accuracy is more important. Using the terminology of normative rationality introduced in (Good, 1952), the medical application of the $\mathcal{TA}$**3** system follows type I rationality, i.e., inference that is consistent with the axioms of decision theory regardless of the cost of inference. On the other hand, the robotic application of $\mathcal{TA}$**3** implements type II rationality, since the cost of reasoning is considered and the time resources are limited. Thus, even an approximate solution provided in real time is more valuable than an accurate solution delivered late. In the inverse kinematics task, there are available fast techniques that produce an accurate solution from an approximate one. However, for some robotic architectures, there might not be a computational solution to the problem. Thus, the main objective is to have a solution available within given real time constraints.

Complex information systems for diverse domains need a number of advanced facilities. These include decision support systems (DSSs), repositories, reasoning systems, and facilities for modeling and processing inter-related information. CBR systems are an important class of DSSs. They require a design process that systematically produces systems of high quality. Beyond satisfying functional requirements for CBR, it is important to meet global quality factors, such as performance and confidentiality, called non-functional requirements (NFRs). In (Jurisica and Nixon, 1998) we present a goal-oriented, knowledge-based approach for aiding DSS development and usage, namely, it proposes an approach for dealing with NFRs for CBR systems. We show how quality can be built into a CBR system, using the "QualityCBR" approach, which integrates existing work on CBR and NFRs. We illustrate the use of the approach on an *in vitro* fertilization domain. The QualityCBR approach is used to address important NFRs, such as performance, accuracy and confidentiality.

$\mathcal{TA}$3 differs from other case-based approaches because it does not use predefined retrieval strategies. Instead, case retrieval is custom-tailored, and dynamically changed for a particular domain and specific application. $\mathcal{TA}$**3** can be easily configured for individual tasks and it allows for flexibility in certain aspects of computation, such as trading off the accuracy or precision of the computation process for time resources. The advantages of the $\mathcal{TA}$**3** system include: (1) added semantic meaning (in the form of context) to similarity and thus to retrieval and explanation of the inference process; (2) meaningful classification, which improves access time and makes visualization easier to understand; and (3) knowledge mining is user-oriented and enables one to control precision, recall and coverage.

Although the similarity-based retrieval mechanism is part of the $\mathcal{TA}$3 CBR system, the application of the underlying algorithm to retrieval from a software repository shows that variable-context similarity assessment is useful in domains where information can be represented as attribute-value

pairs. Thus, similarity-based retrieval tools can be used in building flexible retrieval and classification systems. Variable-context similarity assessment supports flexibility by allowing for different relevancy measures, different retrieval strategies, automatic context relaxation/restriction, and by supporting similarity-based queries, query-by-example and query-by-reformulation (Jurisica, 1996).

8.2 Future Work

There are several possibilities for future work. First of all, the prototype needs a more robust implementation. The system should be implemented as a "pluggable" tool, so it can easily interface with other tools and systems. In addition, it should be allowed to change case representation flexibly and dynamically, instead of using a fixed representation schema, which can only be changed via re-compilation. This is inline with the proposed future AI paradigm (Hayes-Roth, 1997)

Additional flexibility and efficiency gains are possible if an off-the-shelf database management system for storing cases and domain knowledge is used. This would allow for employing a more complex case organization and utilizing multimedia information, which is necessary in complex domains (Jurisica et al., 1998; Jurisica and Gupta, 1997).

More robust knowledge-mining and machine-learning algorithms would potentially help in creating an initial case base by selecting only relevant case descriptors, and by evolving the case representation schema to accommodate knowledge evolution and environmental changes. It is not a trivial task to create an effective case base. Some approaches use a simplistic data representation to make the acquisition process easier (Kitano, Shibata and Shimazu, 1993). Others modify the retrieval algorithm to suit information acquisition needs (Mookerjee and Mannino, 1997). Alternatively, case base engineering tools can help to create case bases from existing information bases of diverse types (Gupta, 1997).

From the user point of view, a simpler query-language with visual query transformation would be desirable. Introducing user modeling capabilities would allow for selecting preferences suited to a specific user. By having a hierarchy of such user models, the system could change from passive to pro-active in suggesting the best ways to obtain information. In addition, combining user modeling and natural language processing (as in FAQ Finder (Burke et al., 1997)), a system may present the same information to diverse users in a different format (Hirst et al., 1997).

System efficiency can be further improved by using a caching mechanism that takes advantage of incremental context transformations during iterative browsing. This enables an incremental evaluation of new queries, based on previous evaluations. However, such an approach would require a change-propagation algorithm, since we can no longer assume that the content of the case base remains unchanged in between queries.

Bibliography

References

Acker, L. and Porter, B. (1992). Extracting viewpoints from multifunctional knowledge bases. Technical report, Dept. of Computer Science, University of Texas, Austin, TX.

Adams, R. (1993). An experiment in software retrieval. In *Proc. of the 4th European Software Engineering Conference*, Garmisch, Germany.

Adelson, B., Burstein, M. H., Gentner, D., Hammond, K. J., Holyoak, K. J., and Thagard, P. R. (1988). The role of analogy in a theory of problem-solving. In *Proc. of the 10th Annual Conference of the Cognitive Science Society*, pages 298–304, Montreal, Quebec.

Agrawal, R., Imielinski, T., and Swami, A. (1993). Database mining: A performance perspective. *IEEE Transactions on Knowledge and Data Engineering. Special issue on Learning and Discovery in Knowledge-Based Databases*, 5(6):914–925.

Aha, D. (1992a). Tolerating noisy, irrelevant and novel attributes in instance-based learning algorithms. *International Journal of Man - Machine Studies*, 36(2):267–287.

Aha, D., editor (1994). *AAAI Workshop on Case-Based Reasoning*. Seattle, WA.

Aha, D., Kibler, D., and Albert, M. (1991). Instance-based learning algorithms. *Machine Learning*, 6(1):37–66.

Aha, D. W. (1992b). Generalizing from case studies: A case study. In *Proc. of the 9th International Conference on Machine Learning*, pages 1–10, Aberdeen, Scotland.

Aha, D. W. (1995). An implementation and experiment with the nested generalized exemplars algorithm. Technical Report AIC-95-003, Naval Research Laboratory, Navy Center for Applied Research in Artificial Intelligence, Washington, DC.

Aha, D. W. and Bankert, R. L. (1995). A comparative evaluation of sequential feature selection algorithms. In *Proceedings of the Fifth International Workshop on Artificial Intelligence and Statistics*, Ft. Lauderdale, FL.

Aha, D. W. and Breslow, L. A. (1997). Refining conversational case reasoning. In *Proc. of the 2nd International Conference on Case-Based Reasoning*, Providence, RI.

Aha, D. W. and Salzberg, S. L. (1994). Learning to catch: Applying nearest neighbor algorithms to dynamic control tasks. In Cheeseman, P. and Oldford, R. W., editors, *Selecting Models from Data: Artificial Intelligence and Statistics IV*. Springer-Verlag.

Alterman, R. (1986). An adaptive planner. In *Proc. of AAAI-86*, pages 65–69, Philadelphia, PA.

Alterman, R. (1988). Adaptive planning. *Cognitive Science*, 12:393–422.

Arikawa, S., Miyano, S., Shinohara, A., Kuhara, S., Mukouchi, Y., and Shinohara, T. (1993). A machine discovery from amino acid sequences by decision trees over regular patterns. *New Generation Computing*, 11(3-4):361–375.

Arimoto, S., Kawamura, S., and Miyazaki, F. (1984). Bettering operation for robots by learning. *Journal of Robotic Systems*, (1):123–140.

Ashley, K. D. (1989). Assessing similarities among cases. In *Proc. of 2 nd Workshop on CBR*, pages 72–76, Pensacola Beach, FL.

Ashley, K. D. (1990). *Modeling Legal Argument: Reasoning with Case and Hypotheticals*. MIT Press/Bradford Books, Cambridge, MA.

Astakhov, Y., Zubanov, K., Kavchenkov, V., and Pashenkova, T. (1993). Application of similarity theory in forecasting electrical energy generation. *Electrical Technology*, (1):139–154.

Attardi, G. and Simi, M. (1993). A formalization of viewpoints. Technical Report TR-93-062, International Computer Science Institute, Berkeley, CA.

Bækgaard, L. and Mark, L. (1995). Incremental computation of time-varying query expressions. *IEEE Transactions on Knowledge and Data Engineering*, 7(4):583–589.

Bailey, D., Thompson, D., and Feinstein, J. (1987). Similarity networks as a knowledge representation for space application. In *Proc. of 3rd Conference on Artificial Intelligence for Space Application*, pages 279–284, Huntsville, AL.

Bain, A. (1855). *Senses of Intellect*. J.W. Parker, London, UK.

Balakrishnan, S. and Weil, R. (1996). Neurocontrol: A literature survey. *Mathematical and Computer Modelling*, 23(1-2):101–117.

Bancilhon, F. (1986). Naive evaluation of recursively defined relations. In Brodie, M. L. and Mylopoulos, J., editors, *On knowledge base management systems: Integrating artificial intelligence and database technologies*, pages 165–178. Springer-Verlag, New York, NY.

Bar, M. and Ullman, S. (1993). Spatial context in recognition. Technical Report 22, The Weizmann Institute of Science, Dept. of Applied Mathematics and Computer Science, Rehovot, Israel.

Bareiss, E. and King, J. A. (1989). Similarity assessment in case-based reasoning. In *Proc. of 2 nd Workshop on CBR*, pages 67–71, Pensacola Beach, FL.

Bareiss, E., Porter, B. W., and Wier, C. (1988). The exemplar-based learning apprentice. Technical Report AI87-53, The University of Texas at Austin.

Bareiss, E. R. (1988). *A unified approach to concept representation, classification, and learning.* PhD thesis, Department of Computer Science, University of Texas.

Bareiss, E. R. (1989a). *Exemplar-Based Knowledge Acquisition: A Unified Approach to Concept Representation, Classification, and Learning.* Academic Press, Boston, MA.

Bareiss, E. R. (1989b). The experimental evaluation of a case-based learning apprentice. In *Proc. of 2 nd Workshop on CBR*, pages 162–167, Pensacola Beach, FL.

Barletta, R. and Hennessy, D. (1989). Case adaptation in autoclave layout design. In *Proc. of 2 nd Workshop on CBR*, pages 203–207, Pensacola Beach, FL.

Barletta, R. and Mark, W. (1988). Explanation-based indexing of cases. In *Proc. of AAAI-88*, pages 541–546, St. Paul, MN.

Barnden, J. A. and Holyoak, K. J. (1994). *Analogy, Metaphor, and Reminding*, volume 3. Ablex, Norwood, N.J.

Barsalou, L. W. (1989). Intraconcept similarity and its implications for interconcept similarity. In *Similarity and Analogical Reasoning*, pages 76–121, New York. Cambridge University Press.

Barwise, J. (1989). *The Situation in Logic.* Center for the Study of Language and Information.

Belkin, N. J., Cool, C., Stein, A., and Thiel, U. (1995). Case, scripts, and information-seeking strategies: On the design of interactive information retrieval systems. *Expert Systems with Applications*, 9(3):379–395.

Bencke (1871). In (Gregson, 1975).

Bento, C. and Costa, E. (1994). *A similarity metric for retrieval of cases imperfectly explained.* Springer Verlag.

Bergmann, R. and Wilke, W. (1995). Building and refining abstract planning cases by change of representation language. *Journal of Artificial Intelligence Research*, 3:53–118.

Berman, D. H. and Hafner, C. D. (1993). Representing teleological structure in case-based legal reasoning: The missing link. In *Proc. of the 4rd International Conference on Artificial Intelligence and Law*, Amsterdam, The Netherlands.

Bertino, E. (1992). A view mechanism for object-oriented databases. In *Int. Conference on Extending Database Technologies*, Vienna, Austria.

Bhansali, S. and Harandi, M. (1993). Synthesis of UNIX programs using derivational analogy. *Machine Learning*, 10(1):7–55.

Binaghi, E., Della Ventura, A., Rampini, A., and Schettini, R. (1993). Fuzzy reasoning approach to similarity evaluation in image analysis. *International Journal of Intelligent Systems*, 8:749–769.

Birnbaum, L. and Collins, G. C. (1988). The transfer of experience across planning domains through the acquisition of abstract strategies. In *Proc. of 1st Workshop on CBR*, pages 61–79, Clearwater Beach, FL.

Birnbaum, L. and Collins, G. C. (1989). Remindings and engineering design themes: A case study in indexing vocabulary. In *Proc. of 2 nd Workshop on CBR*, pages 47–51, Pensacola Beach, FL.

Birnbaum, L., Collins, G. C., Freed, M., and Krulwich, B. (1990). Model-based diagnosis of planning failures. In *Proc. of AAAI-90*, pages 318–323, Boston, MA.

Blakeley, J. A., Larson, P.-A., and Tompa, F. W. (1986). Efficiently updating materialized views. In *ACM-SIGMOD*, pages 61–71.

Bondi, P., Casalino, G., and Gambardella, L. (1988). On the iterative learning control theory for robotic manipulators. *IEEE Journal of Robotics and Automation*, 4(1):14–22.

Boström, H. and Idestam-Almquist, P. (1989). APPRENTICE88 - A case-based refiner for heuristic classification systems. Technical Report 157, Department of Computer and Systems Sciences, University of Stockholm, Stockholm, Sweden.

Brachman, R. J., Selfridge, P. G., Terveen, L. G., Altman, B., Borgida, A., Halper, F., Kirk, T., Lazar, A., McGuinness, D. L., and Resnick, L. A. (1993). Integrated support for data archeology. *International Journal of Intelligent and Cooperative Information Systems*, 2(3):159–185.

Bradtke, S. and Lehnert, W. G. (1988). Some experiments with case-based search. In *Proc. of AAAI-88*, pages 80–93, St. Paul, MN.

Branting, L. K. (1989). Integrating generalizations with exemplar-based reasoning. In *Proc. of the 11th Annual Conference of the Cognitive Science Society*, pages 139–146, Ann Arbor, MI.

Branting, L. K. (1991). *Integrating rules and precedents for classification and explanation: Automating legal analysis.* PhD thesis, University of Texas, Austin.

Branting, L. K. (1992). A case-based approach to problem formulation. In *Proc. of the 14th Annual Conference of the Cognitive Science Society*, pages 726–731, Bloomington, IN.

Brewer, W. F. (1989). Comments on Part III: The activation and acquisition of knowledge. In *Similarity and Analogical Reasoning*, pages 532–545, New York. Cambridge University Press.

Broos, P. and Branting, L. K. (1993). Compositional instance-based acquisition of preference predicates. In *Proc. of AAAI-93 Workshop on CBR*, Washington, DC.

Brown, A. L. (1989). Analogical learning and transfer: What develops? In *Similarity and Analogical Reasoning*, pages 369–412, New York. Cambridge University Press.

Brown, M. (1992). Importing similarity measures into case retrieval using analogue marker passing. In *Proc. of the 10th ECAI*, pages 590–592, Vienna, Austria.

Buchanan, B. G., editor (1995). *Evaluation of knowledge-based systems. Report from a workshop.* National Library of Medicine, Bethesda, MD.

Buchanan, B. G. and Wilkins, D. C. (1993). *Readings in Knowledge Acquisition and Learning.* Morgan Kaufmann Publishers, San Mateo, CA.

Burke, R. D., Hammond, K. J., Kulyukin, V., Lytinen, S. L., Tomuro, N., and Schoenberg, S. (1997). *Artificial Intelligence Magazine*, 18(2):57–66.

Burstein, M. H. (1986). Concept formation by incremental analogical reasoning and debugging. In Michalski, R. S., Carbonell, J. G., and Mitchell, T. M., editors, *Machine Learning: An Artificial Intelligence Approach*, volume 2, Los Altos, CA. Morgan Kaufmann.

Buvač, S., Buvač, V., and Mason, I. A. (1994). Mathematics of context. Technical report, Computer Science Department, Stanford University, Stanford, CA.

Canfora, G., Cimitile, A., and Munro, M. (1996). An improved algorithm for identifying objects in code. *Software Practice and Experience*, 26(1):25–48.

Carbonell, J. G. (1981). A computational model of analogical problem solving. In *Proc. of the 7th IJCAI*, pages 147–152, Vancouver, Canada.

Carbonell, J. G. (1983). Derivational analogy and its role in problem solving. In *Proc. of AAAI-83*, pages 64–69, Washington, DC.

Carbonell, J. G. (1986). Learning by analogy: Formulating and generalizing plans from past experience. In Michalski, R. S., Carbonell, J. G., and Mitchell, T. M., editors, *Machine Learning: An Artificial Intelligence Approach*, volume 2.

Carbonell, J. G. and Veloso, M. M. (1988). Integrating derivational analogy into a general problem solving architecture. In *Proc. of 1st Workshop on CBR*, pages 104–123, Clearwater Beach, FL.

Cardie, C. (1993a). A case-based approach to knowledge acquisition for domain-specific sentence analysis. In *Proc. of AAAI-93*, Washington, DC. Additional information from personal communication.

Cardie, C. (1993b). Using decision trees to improve case-based learning. In *Proc. of the 10th International Conference on Machine Learning*, Amherst, MA.

Cardie, C. (1996). Automating feature set selection for case-based learning of linguistic knowledge. In *Proc. of the Conference on Empirical Methods in Natural Language Processing*, pages 113–126.

Carnap, R. (1967). *The logical structure of the world. Pseudoproblems in philosophy.* Routledge and Kegan Paul, London, UK.

Carpineto, C. and Romano, G. (1996). A lattice conceptual clustering system and its application to browsing retrieval. *Machine Learning*, 24(2):95–122.

Ceri, S. and Widom, J. (1991). Deriving production rules for incremental view maintenance. In *VLDB-91*, pages 577–589, Barcelona, Spain.

Chee, Y. (1993). Applying Gentner's theory of analogy to the teaching of computer programming. *International Journal of Man - Machine Studies*, 38(3):347–368.

Cheeseman, P., Kelly, J., Self, M., Stutz, J., Taylor, W., and Freeman, D. (1988). Autoclass: A Bayesian classification system. In *Proc. of the 5th International Conference on Machine Learning*, pages 54–64, Ann Arbor, MI.

Cheng, Y. (1991). Context-dependent similarity. *Uncertainty in Artificial Intelligence 6*, 12:41–47.

Chi, M., Feltovich, P., and Glaser, R. (1981). Categorization and representation of physics problems by experts and novices. *Cognitive Science*, 5:121–152.

Chu, W. W., Yang, H., Chiang, K., Minock, M., Chow, G., and Larson, C. (1996). CoBase: A scalable and extensible cooperative information system. *Journal of Intelligent Information Systems*, 6.

Chung, L., Nixon, B. A., Mylopoulos, J., and Yu, E. (1998). *Non-Functional Requirements in Software Engineering.* In preparation.

Cohen, P. R. (1989). Evaluation and case-based reasoning. In *Proc. of 2 nd Workshop on CBR*, pages 168–172, Pensacola Beach, FL.

Collins, A. and Burstein, M. (1989). Afterword: Comments on Parts I, II, and III: A framework for a theory of comparison and mapping. In *Similarity and Analogical Reasoning*, pages 546–566, New York. Cambridge University Press.

Collins, A. M. and Michalski, R. S. (1989). The logic of plausible-reasoning: A core theory. *Cognitive Science*, 13:1–49.

Collins, G. C. (1989). Plan adaptation: A transformational approach. In *Proc. of 2 nd Workshop on CBR*, pages 90–93, Pensacola Beach, FL.

Collins, G. C. and Birnbaum, L. (1990). Problem-solver state descriptions as abstract indices for case retrieval. In *AAAI Spring Symposium on Case-Based Reasoning*, pages 32–35.

Constantopoulos, P., Jarke, M., Mylopoulos, J., and Vassiliou, Y. (1994). The software information base: A server for reuse. Technical Report DKBS-TR-94-1, University of Toronto, Department of Computer Science, Toronto, ONT.

Constantopoulos, P. and Pataki, E. (1992). A browser for software reuse. In Loucopoulos, P., editor, *Advanced Information Systems Engineering: Proc. of the 4th International Conference CAiSE'92*, pages 304–326. Springer, Berlin, Heidelberg.

Converse, T., Hammond, K. J., and Marks, M. (1989). Learning modification rules from expectation failure. In *Proc. of 2 nd Workshop on CBR*, pages 110–114, Pensacola Beach, FL.

Cook, D. J. (1991). The base selection task in analogical planning. In *Proc. of the 12th IJCAI*, pages 790–795, Sydney, Australia.

Cook, D. J. and Holder, L. B. (1994). Substructure discovery using minimum description length and background knowledge. *Journal of Artificial Intelligence Research*, 1.

Cox, M. T. and Ram, A. (1992). An explicit representation of forgetting. In *Proc. of the 6th International Conference on Systems Research, Informatics and Cybernetics*, Baden-Baden, Germany.

Cunningham, P., Bonzano, A., and Smyth, B. (1995). An incremental case retrieval mechanism for diagnosis. Technical Report TCD-CS-95-01, Trinity College, Dublin, Ireland.

Daudjee, K. S. and Toptsis, A. A. (1994). Automatic organization of reusable software components in a multidimensional space. In De, P. and Woo, C., editors, *4th Annual Workshop on Information Technologies and Systems*, pages 11–20, Vancouver, BC.

Davies, G. M. and Thomson, D. M., editors (1988). *Memory in Context: Context in Memory.* Ellis Horwood, Chichester, UK.

Davis, O. K. and Rosenwaks, Z. (1995). In vitro fertilization. In Adashi, E. Y., Rock, J. A., and Rosenwaks, Z., editors, *Reproductive Endocrinology, Surgery, and Technology*, volume 2, pages 2319–2334, Philadelphia, PA. Lippencott-Raven Publishers.

Dean, T. and Boddy, M. An analysis of time-dependent planning. In *Proc. of AAAI-88*, pages 49–54, St. Paul, MN.

DeJong, G. F. and Mooney, R. J. (1986). Explanation-based learning: An alternative view. *Machine Learning*, 1(2):145–176.

Deraedt, L. and Bruynooghe, M. (1992). Interactive concept-learning and constructive induction by analogy. *Machine Learning*, 8(2):107–150.

Diab, M. (1992). Software reuse repository. In *Proc. of the WISR-92.*

Díaz, R. P. and Freeman, P. (1987). Classifying software for reusability. *IEEE Software*, 4(16):6–16.

Dierbach, C. and Chester, D. L. (1991). A formal basis for analogical reasoning. In *Proc. of 2nd Int. Conference on Principles of Knowledge Representation and Reasoning.*

Domeshek, E. A. (1992). *Do the right thing: A component theory for indexing stories as social advice.* PhD thesis, Yale University. Also available as a Technical Report 92-26 from Northwestern University.

Dubois, D. and Prade, H. (1994). Similarity-based approximate reasoning. In *Proc. of the IEEE Symposium on Computational Intelligence. Imitating Life*, Orlando, Florida.

Edelman, S. (1993). Representation, similarity, and the chorus of prototypes. Technical Report 10, The Weizmann Institute of Science, Dept. of Applied Mathematics and Computer Science, Rehovot, Israel.

Edmondson, W. H. and Meech, J. F. (1994). Putting task analysis into context. *SIGHI*, 26(4):59–63.

Eskridge, T. (1994). A hybrid model of continuous analogical reasoning. *Advances in Connectionist and Neural Computation Theory, Vol 2*, 2:207–246.

Falkenhainer, B., Forbus, K. D., and Gentner, D. (1989). Structure-maping engine. *Artificial Intelligence*, 41:1–63.

Faustle, S., Fugini, M. G., and Damiani, E. (1996). Retrieval of reusable components using functional similarity. *Software Practice and Experience*, 26(5):491–530.

Féret, M. P. and Glasgow, J. I. (1993). Hybrid case-based reasoning for the diagnosis of complex devices. In *Proc. of AAAI-93*, Washington, DC.

Fernandez-Chamizo, C., Gonzalezcalero, P., Hernandez-Yanez, L., and Urechbaque, A. (1995). Case-based retrieval of software components. *Expert Systems with Applications*, 9(3):397–405.

Fertig, S. and Gelertner, D. H. (1991). FGP: A virtual machine for acquiring knowledge from cases. In *Proc. of the 12th IJCAI*, pages 796–802, Sydney, Australia.

Finnigan, P. J., Holt, R., Kalas, I., Kontogiannis, K., Muller, H. A., Mylopoulos, J., Perelgut, S., Stanley, M., Wong, K., and Kerr, S. (1997). The software bookshelf. *IBM Systems Journal*. Under revisions.

Fisher, D. (1987). Knowledge acquisition via incremental concept clustering. *Machine Learning*, 2(1):39–172.

Forbus, K. D., Gentner, D., and Law, K. (1995). MAC/FAC: A model of similarity-based retrieval. *Cognitive Science*, (19):141–205.

Fouqué, G. and Matwin, S. (1993). Compositional software reuse with case-based reasoning. In *Orlando, FL*, Orlando, FL.

Francis, A. G. and Ram, A. (1993). The utility problem in case-based reasoning. In *Proc. of AAAI-93 Workshop on CBR*, Washington, DC.

Frawley, J. and Piatetsky-Shapiro, G. (1991). *Knowledge Discovery in Databases*. AAAI Press/MIT Press.

Frey, P. W. and Slate, D. J. (1991). Letter recognition using Holland-style adaptive classifiers. *Machine Learning*, 6(2).

Fullerton, G. (1890). *On sameness and Identity*. University of Pennsylvania Press, Philadelphia. PA.

Gaasterland, T. (1993). Restricting query relaxation through user constraints. In *Proc. of International Conference on Intelligent and Cooperative Information Systems*, pages 359–366, Rotterdam, The Netherlands.

Gaasterland, T., Godfrey, P., and Minker, J. (1991). Relaxation as a platform of cooperative answering. In *Proc. International Workshop on Nonstandard Answers and Queries*, Toulouse, France.

Garben, A., Furnsinn, M., and Ruschkowski, B. (1995). Espanda - for solving problems by applying the principle of similarity. *Expert Systems with Applications*, 8(2):249–254.

Garg, P. and Scacchi, W. (1989). ISHYS: Designing an intelligent software hypertext system. *IEEE Expert*, pages 52–82.

Gaschnig, J. (1979). A problem similarity approach to devising heuristics: First results. In *Proc. of AAAI-79*, pages 301–307.

Gentner, D. (1983). Structure-mapping: A theoretical framework for analogy. *Cognitive Science*, 7(2):155–170.

Gentner, D. (1989). The mechanisms of analogical learning. In Vosniadou, S. and Ortony, A., editors, *Similarity and Analogical Reasoning*, pages 199–241, New York. Cambridge University Press.

Gentner, D. and Bowdle, B. F. (1994). The coherence imbalance hypothesis: A functional approach to asymmetry in comparison. In *Proc. of the 16th Annual Conference of the Cognitive Science Society*, pages 351–356, Atlanta, GA.

Gentner, D. and Forbus, K. D. (1991). MAC/FAC: A model of similarity-based retrieval. In *Proc. of the 13th Annual Conference of the Cognitive Science Society*, Chicago, IL.

Gick, M. and Holyoak, K. J. (1980). Analogical problem solving. *Cognitive Psychology*, 12:306–355.

Girardi, M. and Ibrahim, B. (1994). A similarity measure for retrieving software artifacts. Technical report, Univ. of Geneva, Centre Univ. d'Informatique, Geneve, CH.

Goel, A. K. (1989). *Integration of case-based reasoning and model-based reasoning for adaptive design problem solving.* PhD thesis, Department of Computer and Information Science. The Ohio State University.

Goel, A. K. and Chandrasekaran, B. (1989). Use of device models in adaptation of design cases. In *Proc. of 2 nd Workshop on CBR*, pages 100–109, Pensacola Beach, FL.

Golding, A. R. and Rosenbloom, P. S. (1989). Combining analytical and similarity-based CBR. In *Proc. of 2 nd Workshop on CBR*, pages 259–263, Pensacola Beach, FL.

Golding, A. R. and Rosenbloom, P. S. (1991). Improving rule-based systems through case-based reasoning. In *Proc. of AAAI-91*, pages 22–27, Anaheim, CA.

Goldstone, R. and Medin, D. (1994). Similarity, interactive activation, and mapping - An overview. *Advances in Connectionist and Neural Computation Theory, Vol 2*, 2:321–362.

Gonzalez, A. J. and Laureano-Ortiz, R. (1992). A case-based reasoning approach to real estate property appraisal. *Expert Systems with Applications*, 4:229–246.

Good, I. (1952). Relational decisions. *J. R. Stat. Soc. B*, 14:107–114.

Goodman, M. (1989). CBR in battle planning. In *Proc. of 2 nd Workshop on CBR*, pages 264–269, Pensacola Beach, FL.

Goodman, N. (1951). *The Structure of Appearance*. Harvard University Press, Cambridge, MA.

Gravano, L., García-Molina, H., and Tomasic, A. (1994). Precision and recall of *gloss* estimators for database discovery. Technical Report CS-TN-94-10, Stanford University, Department of Computer Science, Stanford, CA.

Green, S. J. (1997). *Automatically generating hypertext by computing semantic similarity*. PhD thesis, University of Toronto, Toronto, ON.

Gregson, R. A. (1975). *Psychometrics of Similarity*. Academic Press, New York, NY.

Greiner, R. (1994). *AAAI Fall Symposium Series on Relevance*. AAAI Press, Menlo Park, CA.

Greiner, R. and Jurisica, I. (1992). A statistical approach to solving the EBL utility problem. In *Proc. of AAAI-92*, pages 241–247, San Jose, CA.

Griffin, T. and Libkin, L. (1995). Incremental maintenance of views with duplicates. In *Proc. of the ACM SIGMOD International Conference on Management of Data*, pages 328–339, San Jose, CA.

Griffiths, A. D. and Bridge, D. G. (1995). On concept space and hypothesis space in case-based learning algorithms. In *Proc. of 8th European Conference on Machine Learning*.

Gupta, A., Mumick, I. S., and Ross, K. A. (1995). Adapting materialized views after redefinitions. In *Proc. of the ACM SIGMOD International Conference on Management of Data*, pages 211–222, San Jose, CA.

Gupta, A., Mumick, I. S., and Subrahmanian, V. (1993). Maintaining views incrementally. In *Proc. of the 12th ACM SIGACT-SIGMOD-SIGART Symposium on Principles of Database Systems*, pages 157–166.

Gupta, K. M. (1997). Case base engineering for large scale industrial applications. In *AAAI Spring Symposium: AI in Knowledge Management*, Stanford, CA.

Hall, R. P. (1989). Computational approaches to analogical reasoning: A comparative analysis. *Artificial Intelligence*, 39:39–120.

Hamilton, H. J. (1990). Datex: A framework for machine discovery of regularity in data. In Patel-Schneider, P. F., editor, *CSCSI-90: Proc. of the 8th Biennial Conference of the Canadian Society for Computational Studies of Intelligence*, pages 196–203. Kaufmann, Palo Alto, CA.

Hammond, K., Burke, R., and Schmitt, K. (1996). A case-based approach to knowledge navigation. In *(Leake, 1996)*, pages 125–136.

Hammond, K. J. (1989a). *Case-Based Planning: Viewing Planning as a Memory Task.* Academic Press, Boston, MA.

Hammond, K. J. (1989b). Opportunistic memory. In *Proc. of the 11th IJCAI*, Detroit, MI.

Hammond, K. J., Burke, R., and Schmitt, K. (1994). Case-based approach to knowledge navigation. In *Proc. of the AAAI Workshop on Knowledge Discovery in Databases*, Seattle, WA.

Hammond, K. J., Converse, T., Marks, M., and Seifert, C. M. (1993). Opportunism and learning. *Machine Learning*, 10(3):279–309.

Hanks, S. and Weld, D. S. (1992). The systematic plan adaptor: A formal foundation for case-based planning. Technical Report 92-09-04, University of Washington, Washington, DC.

Harman, H. (1976). *Modern Factor Analysis.* University of Chicago Press, Chicago, IL.

Hayes-Roth, F. (1997). Artificial intelligence: What works and what doesn't? *Artificial Intelligence Magazine*, 18(2):99–113.

Heckerman, D. (1991). Similarity networks for the construction of multiple-fault belief networks. *Uncertainty in Artificial Intelligence 6*, 12:51–64.

Hennessy, D. and Hinkle, D. (1992). Applying case-based reasoning to autoclave loading. *IEEE Expert*, 7:21–26.

Hinrichs, T. R. (1988). Towards an architecture for open world problem solving. In *Proc. of 1st Workshop on CBR*, pages 182–189, Clearwater Beach, FL.

Hinrichs, T. R. (1989). Strategies for adaptation and recovery in a design problem solver. In *Proc. of 2 nd Workshop on CBR*, pages 115–118, Pensacola Beach, FL.

Hirst, G., , Hovy, E., DiMarco, C., and Parsons, K. (1997). Authoring and generating health-education documents that are tailored to the needs of the individual patient. In *Proceedings of the Sixth International Conference on User Modeling*, Sardinia, Italy.

Holder, L. (1992). Empirical analysis of the general utility problem in machine learning. In *Proc. of AAAI-92*, pages 249–254, San Jose, CA.

Holyoak, K. J. (1985). The pragmatics of analogical transfer. In Bower, G., editor, *The Psychology of Learning and Motivation*, New York, NY. Academic Press.

Holyoak, K. J. and Thagard, P. R. (1989). A computational model of analogical problem solving. In *Similarity and Analogical Reasoning*, pages 242–265, New York. Cambridge University Press.

Hume, D. (1947). An inquiry concerning human understanding. In *The World's Great Thinkers. Man and Spirit: The Speculative Philosophers*, pages 341–423, New York. Random House. Originally in *An inquiry concerning human understanding*, Clarendon Press, 1748.

Hunter, L. (1989). Finding paradigm cases, or when is a case worth remembering? In *Proc. of 2 nd Workshop on CBR*, pages 57–61, Pensacola Beach, FL.

Huuskonen, P. and Korteniemi, A. (1992). Explanation-based on context. In *Proc. of the 8th IEEE Computer Society Conference on Artificial Intelligence Applications*, pages 179–185, Monterey, CA.

Indurkhya, B. (1990). On the role of interpretive analogy in learning. *New Generation Computing*, 8(4):385–402.

Jagadish, H. (1991). A retrieval technique for similar shapes. In *Proc. of the 10th ACM SIGACT-SIGMOD-SIGART Symposium on Principles of Database Systems*, pages 208–217, Denver, CO.

Jagadish, H. V., Mendelzon, A. O., and Milo, T. (1995). Similarity-based queries. In *Proc. of the 14th ACM SIGACT-SIGMOD-SIGART Symposium on Principles of Database Systems*. San Jose, CA.

James, W. (1890). *The Principles of Psychology*. Holt, New York, NY.

Janetzko, D., Wess, S., and Melis, E. (1992). Goal-driven similarity assessment. In Ohlbach, H. J., editor, *Advances in Artificial Intelligence - Proc. of the 16th German Conference on Artificial Intelligence*, pages 283–298, Bonn, Germany.

Jang, T., Choi, C., and Ahn, H. (1995). Iterative learning control in feedback systems. *Automatica*, 31(2):243–248.

Jantke, K. P. (1994). Nonstandard concepts of similarity in case-based reasoning. In *Proc. of the 17th Annual Conference of the Gesellschaft für Klassifikation e.V.*, Kaiserslautern, Germany. Springer Verlag.

Jeng, J.-J. and Cheng, B. H. C. (1993). Using analogy and formal methods for software reuse. In *Proc. of IEEE Conf. on Tools for AI*, pages 113–116.

Jones, E. K. (1989). Case-based analogical reasoning using proverbs. In *Proc. of 2 nd Workshop on CBR*, pages 275–279, Pensacola Beach, FL.

Jones, E. K. (1991). Adapting abstract knowledge. In *Proc. of the 13th Annual Conference of the Cognitive Science Society*, pages 155–160, Chicago, IL.

Jones, E. K. (1992). Model-based case adaptation. In *Proc. of AAAI-92*. pages 673–678, San Jose, CA.

Jurisica, I. (1994). Context-based similarity applied to retrieval of relevant cases. Technical Report DKBS-TR-94-5, Univ. of Toronto, Dept. of Comp. Sci., Toronto, ON.

Jurisica, I. (1995). *TA3* : Case-based intelligent retrieval and advisory tool. In *ACM Conference on Society and the Future of Computing*, Durango, CO.

Jurisica, I. (1996). Supporting flexibility. a case-based reasoning approach. In *The AAAI Fall Symposium. Flexible Computation in Intelligent Systems: Results, Issues. and Opportunities*, Cambridge, Massachusetts.

Jurisica, I. (1997). Similarity-based retrieval for diverse Bookshelf software repository users. In *IBM CASCON Conference*, pages 224–235, Toronto, Canada.

Jurisica, I. and Glasgow, J. (1996a). Case-based classification using similarity-based retrieval. In *8th IEEE Int. Conf. on Tools with Artificial Intelligence*, Toulouse, France.

Jurisica, I. and Glasgow, J. (1996b). A case-based reasoning approach to learning control. In *5th International Conference on Data and Knowledge Systems for Manufacturing and Engineering, DKSME-96*, Phoenix, Arizona.

Jurisica, I. and Glasgow, J. (1997). Improving performance of case-based classification using context-based relevance. *International Journal of Artificial Intelligence Tools. Special Issue of IEEE ITCAI-96 Best Papers*, 6(4):511–536.

Jurisica, I. and Glasgow, J. (1998). An efficient approach to iterative browsing and retrieval for case-based reasoning. In del Pobil, A. P., Mira, J., and Ali, M., editors, *Lecture Notes in Computer Science, IEA/AIE*98*. Springer-Verlag.

Jurisica, I. and Gupta, K. M. (1997). Knowledge-based systems for decision support in health care. In *Digital Knowledge Conference II*, Toronto, Canada.

Jurisica, I., Mylopoulos, J., Glasgow, J., Shapiro, H., and Casper, R. F. (1998). Case-based reasoning in IVF: Prediction and knowledge mining. *Artificial Intelligence in Medicine*, 12(1):1–24.

Jurisica, I. and Nixon, B. (1998). Building quality into case-based reasoning systems. In *CAiSE*98, Lecutre Notes in Computer Science*. Springer-Verlag.

Jurisica, I. and Shapiro, H. (1995). A computer model for case-based reasoning in IVF. In *The 51st Conf. of the American Society for Reproductive Medicine*, Seattle, WA.

Kaelbling, L. (1994). Associative reinforcement learning: A generate and test algorithm. *Machine Learning*, 15(3):299–319.

Kant, I. (1947). The critique of pure reason. In *The World's Great Thinkers. Man and Spirit: The Speculative Philosophers*, pages 423–442, New York. Random House.

Kashyap, V. and Sheth, A. (1993). Schema correspondences between objects with semantic proximity. Technical Report DCS-TR-301, Department of Computer Science, Rutgers University.

Kass, A. M. (1990). *Developing creative hypotheses by adapting explanations.* PhD thesis, Yale University.

Kass, A. M. (1991). Adaptation strategies for case-based plan recognition. In *Proc. of the 13th Annual Conference of the Cognitive Science Society*, pages 161–166, Chicago, IL.

Kass, A. M. and Leake, D. B. (1988). Case-based reasoning applied to constructing explanations. In *Proc. of 1st Workshop on CBR*, pages 190–208, Clearwater Beach, FL.

Keane, M. T. (1988). *Analogical Problem Solving.* Ellis Horwood, Chichester, UK.

Keane, M. T., Ledgeway, T., and Duff, S. (1991). Constraint on analogical mapping: The effects of similarity & order. In *Proc. of the 13th Annual Conference of the Cognitive Science Society*, pages 275–280, Chicago, IL.

Kibler, D. and Aha, D. (1987). Learning representative exemplars of concepts: An initial case study. In *Proc. of the 4th International Workshop on Machine Learning*, pages 24–30, Irvine, CA.

King, J., Klein, G., Whitaker, L., and Wiggins, S. (1988). SURVER III: An application of case-based reasoning. In *Proceedings of the 4th Annual Aerospace Applications of AI Conference*, Dayton, OH.

Kitano, H., Shibata, A., and Shimazu, H. (1993). Case-method: A methodology for building large-scale case-based systems. In *Proc. of AAAI-93*, pages 303–308, Washington, DC.

Kitano, H., Shibata, A., Shimazu, H., Kajihara, J., and Sato, A. (1992). Building large-scale and corporate-wide case-based systems: Integration of organizational and machine executable algorithms. In *Proc. of AAAI-92*, pages 843–849, San Jose, CA.

Kohavi, R. (1997). Data mining using MLC++. *International Journal of Artificial Intelligence Tools. Special Issue of IEEE ITCAI-96 Best Papers*, 6(3, 4). In Press.

Kolodner, J. L. (1983). Maintaining organization in a dynamic long-term memory. *Cognitive Science*, 7(4):281–328.

Kolodner, J. L. (1987). Capitalizing on failure through case-based inference. In *Proc. of the 9th Annual Conference of the Cognitive Science Society*, pages 715–726, Seattle, WA.

Kolodner, J. L. (1989). Selecting the best case for a case-based reasoner. In *Proc. of 2 nd Workshop on CBR*, pages 77–81, Pensacola Beach, FL.

Kolodner, J. L. (1992). An introduction to case-based reasoning. *Artificial Intelligence Review*, 6(1):3–34.

Kolodner, J. L. (1993). *Case-Based Reasoning*. Morgan Kaufmann, San Mateo, CA.

Kolodner, J. L. and Kolodner, R. (1987). Using experience in clinical problem solving: Introduction and framework. *IEEE Transactions on Systems, Man, and Cybernetics*, 17(3):420–431.

Kolodner, J. L. and Riesbeck, C. K. (1986). *Experience, Memory, and Reasoning*. Lawrence Erlbaum Assoc., Inc., Hillsdale, NJ.

Kolodner, J. L. and Simpson, R. L. (1989). The MEDIATOR: Analysis of an early case-based problem solver. *Cognitive Science*, 13(4):507–549.

Kononenko, I. and Bratko, I. (1991). Information-based evaluation criteria for classifier's performance. *Machine Learning*, 6(1):67–80.

Koton, P. (1988a). Reasoning about evidence in causal explanation. In *Proc. of AAAI-88*, pages 256–261, St. Paul, MN.

Koton, P. (1988b). *Using experience in learning and problem solving*. PhD thesis, Computer Science Department, MIT.

Krovvidy, S. and Wee, W. G. (1993). Wastewater treatment systems from case-based reasoning. *Machine Learning*, 10(3):341–363.

Külpe, O. (1895). *Outlines of Psychology*. Swan Sonnenschein, London, UK.

Kuniyoshi, Y., Inaba, M., and Inoue, H. (1994). Learning by watching: Extracting reusable task knowledge from visual observation of human performance. *IEEE Transaction on Robotics and Automation*, 10(6):799–822.

Langley, P., Simon, H. A., and Bradshaw, G. L. (1990). Heuristics for empirical discovery. In Shavlik, J. W. and Dietterich, T. G., editors, *Readings in Machine Learning*, pages 356–372. Kaufmann, San Mateo, CA.

Lauzon, D. and Rose, T. (1994). Task-oriented and similarity-based retrieval. In *Proc. of 9th Conference on Knowledge-Based Software Engineering*, Monterey, CA.

Law, K., Forbus, K. D., and Gentner, D. (1994). Simulating similarity-based retrieval: A comparison of ARCS and MAC/FAC. In *Proc. of the 16th Annual Conference of the Cognitive Science Society*, pages 543–548, Atlanta, GA.

Leake, D. (1992a). *Evaluating explanations: A content theory.* Lawrence Erlbaum, Hillsdale, NJ.

Leake, D. (1995). Adaptive similarity assessment for case-based explanation. *International Journal of Expert Systems*, 8(2):165–194.

Leake, D., editor (1996). *Case-Based Reasoning: Experience, lessons, and future directions.* AAAI Press.

Leake, D., Kinley, A., and Wilson, D. (1997). Case-based similarity assessment: Estimating adaptability from experience. In *Proc. of AAAI-97.*

Leake, D. B. (1991). ACCEPTER: A program for dynamic similarity assessment in case-based explanation. In *Proc. of 4th Workshop on CBR*, pages 51–62, Washington, D.C.

Leake, D. B. (1992b). Constructive similarity assessment: Using stored cases to define new situations. In *Proc. of the 14th Annual Conference of the Cognitive Science Society*, pages 313–318, Bloomington, IN.

Lebowitz, M. (1986). Not the path to perdition: The utility of similarity-based learning. In *Proc. of AAAI-86*, pages 533–537, Philadelphia, PA.

Lebowitz, M. (1987). Experiments with incremental concept formation. *Machine Learning*, 2(1):103–138.

Lee, C. (1994). An information theoretic similarity-based learning method for databases. In *Proc. of the 10th IEEE Computer Society Conference on Artificial Intelligence Applications*, pages 99–105, San Antonio, TX.

Lee, H.-Y. and Harandi, M. T. (1993). An analogy-based retrieval mechanism for software design reuse. In *Proc. of the 8th Knowledge-Based Software Engineering Conference*, pages 152–159, Chicago, IL.

Leng, B., Buchanan, B. G., and Nicholas, H. B. (1993). Protein secondary structure prediction using two-level case-based reasoning. In Searls, D. and Shavlik, J., editors, *Proc. of the 1st International Conference on Intelligent Systems for Molecular Biology*, Washington, DC.

Levy, A. Y. (1993). *Irrelevance reasoning in knowledge based systems.* PhD thesis, Stanford, CA.

Levy, A. Y. (1994). Creating abstractions using relevance reasoning. In *Proc. of AAAI-94*, Seattle, WA.

Levy, A. Y. and Sagiv, Y. (1993). Exploiting irrelevance-reasoning to guide problem-solving. In *Proc. of the 13th IJCAI*, Chambery, France.

Li, X., Hall, N., and Humphreys, G. (1993). Discrete distance and similarity measures for pattern candidate select ion. *Pattern Recognition*, 26(6):843–851.

Liang, T. (1993). Analogical reasoning and case-based learning in model management systems. *Decision Support Systems*, 10(2):137–160.

Liang, T. and Konsynski, B. (1993). Modeling by analogy - use of analogical reasoning in model management systems. *Decision Support Systems*, 9(1):113–125.

Light, P. and Butterworth, G., editors (1993). *Context and Cognition: Ways of Learning and Knowing.* L. Erlbaum Associates, Hillsdale, NJ.

Lipski, W. (1979). On semantic issues connected with incomplete information databases. *ACM Transactions on Database Systems*, 4(3):262–296.

Lockhart, R. S. (1988). Conceptual specificity in thinking and remembering. In Davies, G. M. and Thomson, D. M., editors, *Memory in Context: Context in Memory*, Chichester, UK. Ellis Horwood.

Louis, S., McGraw, G., and Wyckoff, R. O. (1992). CBR assisted explanation of GA results. Technical Report 361, Department of Computer Science, Indiana University, Bloomington, IN. Also in the Journal of Theoretical and Experimental Artificial Intelligence, 5, 1993, pages 21-37.

Lowe, D. G. (1993). Similarity metric learning for a variable-kernel classifier. Technical Report TR-93-43, Univ. of British Columbia.

Luzeaux, D. and Zavidovique, B. (1994). Process control and machine learning: Rule-based incremental control. *IEEE Transactions on Automatic Control*, 39(6):1166–1171.

Maarek, Y., Berry, D., and Kaiser, G. (1991). An information retrieval approach to automatically constructing software libraries. *IEEE Transaction on Software Engineering*, 17(8):800–813.

Maher, P. (1993). A similarity measure for conceptual graphs. *International Journal of Intelligent Systems*, 8(8):819–837.

Martin, T. P., Hung, H.-K., and Walmsley, C. (1992). Supporting browsing of large knowledge bases. Technical report, Department of Computing and Information Science, Queen's University, Kingston, Ontario.

Matheus, C. J., Chan, P. K., and Piatetsky-Shapiro, G. (1993). Systems for knowledge discovery in databases. *IEEE Transactions on Knowledge and Data Engineering*, 5(6):903–913.

McCarthy, J. (1958). Programs with common sense. In *Proc. of the Symposium on the Mechanisation of Thought Precesses*, pages 77–84, National Physical Laboratory.

McCarthy, J. (1993). Notes on formalizing context. In *Proc. of the 13th IJCAI*, pages 555–560, Chambery, France.

McCarthy, J. and Buvač, S. (1994). Formalizing context. Technical Report CS-TN-94-13, Stanford University, Computer Science Department, Stanford, CA.

McDougal, T. F., Hammond, K. J., and Seifert, C. M. (1991). A functional perspective on reminding. In *Proc. of the 13th Annual Conference of the Cognitive Science Society*, pages 510–515, Chicago, IL.

Medin, D. and Ortony, A. (1989). Comments on Part I: Psychological essentialism. In *Similarity and Analogical Reasoning*, pages 179–195. Cambridge University Press.

Michalski, R. S. (1993). Inferential theory of learning as a conceptual basis for multistrategy learning. *Machine Learning*, 11(2):3–151.

Michalski, R. S., Carbonell, J. G., and Mitchell, T. M. (1986). *Machine Learning: An Artificial Intelligence Approach*, volume 2. Morgan Kaufmann, Palo Alto, CA.

Mill, J. (1829). *Analysis of the Phenomena of the Human Mind.* Baldwin and Cradock, London, UK.

Milosavljevic, A. and Jurka, J. (1993). Discovery by minimal length encoding - a case study in molecular evolution. *Machine Learning*, 13(1):153.

Minton, S. N. (1988a). *Learning search control knowledge: An explanation-based approach.* Kluwer Academic Publishers, Hingham, MA.

Minton, S. N. (1988b). Quantitative results concerning the utility of explanation-based learning. In *Proc. of AAAI-88*, pages 564–569, St. Paul, MN. Also published in Journal of Artificial Intelligence, 42, 1990, pages 363-381.

Minton, S. N., Carbonell, J. G., Knoblock, C., Kuokka, D., Etzioni, O., and Gil, Y. (1989). Explanation-based learning: A problem solving perspective. *Artificial Intelligence*, 40:63–118.

Mitchell, T. M. (1990). Becoming increasingly reactive. In *Proc. of the Workshop on Innovative Approaches to Planning, Scheduling and Control*, San Diego, CA.

Mitrovič, A., Witten, I. H., and Maulsby, D. L. (1994). An experiment in the application of similarity-based learning to programming by example. *International Journal of Intelligent Systems*, 9(4):341–364.

Mittermeir, R. T., Mili, R., and Mili, A. (1993). Building a repository of software components: A formal specification approach. In 6^{th} *Annual Workshop on Software Reuse*.

Mookerjee, V. S. and Mannino, M. V. (1997). redesigning case retrieval to reduce information acquisition costs. *Information Systems Research*, 8(1):51–68.

Mooney, R. (1989). The effect of the rule use on the utility of explanation-based learning. In *Proc. of the* 11^{th} *IJCAI*, pages 725–730, Detroit, MI.

Mosteller, F. and Tukey, J. W. (1977). *Data Analysis and Regression*. Addison-Wesley.

Motschnig-Pitrik, R. (1995). An integrating view on the viewing abstraction: Contexts and perspectives in software development, AI and databases. *The Journal of Systems Integration*, 5(1).

Murphy, G. L. and Medin, D. L. (1985). Role of theories in conceptual coherence. *Psychological Review*, (92):289–316.

Mylopoulos, J., Borgida, A., Jarke, M., and Koubarakis, M. (1990). Telos: Representing knowledge about information systems. *ACM Transactions on Information Systems*, 8(4):325–362.

Mylopoulos, J. and Motschnig-Pitrik, R. (1995). Partitioning information bases with contexts. In *Proc. of the* 3^{rd} *International Conference on Cooperative Information Systems*, Vienna, Austria.

Nakamura, K., Sage, A. P., and Iwai, S. (1983). An intelligent data-base interface using psychological similarity between data. *IEEE Transaction on Systems, Man, and Cybernetics*, 13(4):558–568.

Nakamura, K., Sage, A. P., and Iwai, S. (1984). Associative learning using similarity knowledge bases for relational database search. *Future Generation Computer Systems*, 1(2):123–133.

Naniwa, T. and Arimoto, S. (1995). Learning control for robot tasks under geometric endpoint constraints. *IEEE Transactions on Robotics and Automation*, 11(3):432–441.

Navinchandra, D. (1988). Case based reasoning in CYCLOPS, a design problem solver. In *Proc. of* 1^{st} *Workshop on CBR*, pages 286–301, Clearwater Beach, FL.

Navinchandra, D., Sycara, K. P., and Narasimhan, S. (1991). Behavioral synthesis in CADET, a case-based design tool. In *Proc. of the 7th IEEE Computer Society Conference on Artificial Intelligence Applications*, pages 217–221, Miami, CA.

Nayak, P. P., Joskowicz, L., and Addanki, S. (1990). Automated model selection using context-dependent behaviors. Technical Report KSL 90-65, Stanford University, Knowledge Systems Laboratory, Stanford, CA.

Nelson, G., Thagard, P., and Hardy, S. (1994). Integrating analogy with rules and explanations. *Advances in Connectionist and Neural Computation Theory, Vol 2*, 2:181–206.

Nosofsky, R. M. (1990a). Exemplars, prototypes, and similarity rules. Technical Report 17, Indiana University, Department of Psychology, Bloomington, IN.

Nosofsky, R. M. (1990b). Stimulus bias, asymmetric similarity, and classification. Technical Report 14, Indiana University, Department of Psychology, Bloomington, IN.

Ohnishi, H., Suzuki, H., and Shigemasu, K. (1994). Similarity by feature creation: Reexamination of the asymmetry of similarity. In *Proc. of the 16th Annual Conference of the Cognitive Science Society*, pages 687–692, Atlanta, GA.

O'Leary, D. E. (1993). Verification and validation of case-based systems. *Expert Systems with Applications*, 6(1):57–66.

Ortega, J. (1995). On the informativeness of the DNA promoter sequences domain theory. *Journal of Artificial Intelligence Research*, 2:361–367. Research Note.

Ostertag, E., Hendler, J., Díaz, R. P., and Braun, C. (1992). Computing similarity in a reuse library system: An AI-based approach. *ACM Transactions on Software Engineering and Methodology*, 3(1):205–228.

Owens, C. (1989). Plan transformations as abstract indices. In *Proc. of 2nd Workshop on CBR*, pages 62–65, Pensacola Beach, FL.

Owens, C. (1993). Integrating feature extraction and memory search. *Machine Learning*, 10(3):311–339.

Owens, C. (1994). Retriever and Anon: Retrieving structures from memory. In Schank, R., Riesbeck, C., and Kass, A., editors, *Inside Case-Based Reasoning*, pages 89–126. Lawrence Erlbaum.

Pankakoski, J., Eloranta, E., Luhtala, M., and Nikkola, J. (1991). Applying case-based reasoning to manufacturing systems design. *Computer Integrated Manufacturing Systems*, 4(4):212–220.

Parsaye, K., Chignell, M., Khoshafian, S., and Wong, H. (1991). Intelligent data base and automatic discovery. *Neural and Intelligent Systems Integration*, pages 615–628.

Pavlov, I. (1927). *Conditioned Reflexes*. Oxford University Press, London, UK.

Pearce, M., Goel, A., Kolodner, J., Zimring, C., Santosa, L., and Billington, R. (1992). Case-based design support - A case study in architectural design. *IEEE Expert*, 7(5):14–20.

Peterson, L. L. (1979). Semantic similarity analysis - a computer-based study of meaning in noun phrases. In *Proceedings of the AFIPS National Computer Conference*, pages 1063–1070, New York, New York.

Plaza, E., Arcos, J. L., , and Martin, F. (1996). Cooperation modes among case-based reasoning agents. In *Workshop on Learning in Distributed AI Systems, ECAI-96*, Budapest, Hungary.

Porter, B. W. (1989). Similarity assessment: Computation *vs.* representation. In *Proc. of 2 nd Workshop on CBR*, pages 82–84, Pensacola Beach, FL.

Porter, B. W., Bareiss, E., and Holte, R. (1989). Knowledge acquisition and heuristic classification in weak-theory domains. Technical Report AI89-96, Artificial Intelligence Laboratory, University of Texas, Austin.

Porter, B. W., Bareiss, E., and Holte, R. (1990). Concept learning and heuristic classification in weak-theory domains. *Artificial Intelligence*, 45:229–263.

Porter, B. W., Lester, J. C., Murray, K., Pittman, K. E., Souther, A., Acker, L., and Jones, T. (1988). AI research in the context of multifunctional knowledge base: The botany knowledge base project. Technical Report AI-TR-88-88, Department of Computer Sciences, University of Texas, Austin, TX.

Portinale, L. (1991). Generalization handlig in a dynamic case memory. In Z.W.Raś and Zemankova, M., editors, *Proc. of the 6th International Symposium on Methodologies for Intelligent Systems*, pages 72–81, Charlotte, N.C.

Prakash, V. (1994). Structural similarity among proteins from oilseeds: An overview. *Journal of Scientific & Industrial Research*, 53(9):684–691.

Quinlan, J. R. (1986a). The effect of noise in concept learning. In Michalski, R. S., Carbonell, J. G., and Mitchell, T. M., editors, *Machine Learning: An Artificial Intelligence Approach*, pages 149–166, Los Altos, CA. Morgan Kaufmann.

Quinlan, J. R. (1986b). Induction of decision trees. *Machine Learning*, 1(1):81–106.

Quinlan, J. R. (1992). Learning with continuous classes. In *Proc. 5th Australian Joint Conference on AI*, pages 343–348.

Quinlan, J. R. (1993). Combining instance-based and model-based learning. In *Proc. of the 10th International Conference on Machine Learning*, Amherst, MA.

Rafiei, D. and Mendelzon, A. (1997). Similarity-based queries for time series data. In *Proc. ACM SIGMOD. International Conference on Management of Data*, Tucson, AZ.

Ram, A. (1993a). AQUA: Questions that drive the explanation process. In *Inside Computer Explanation*. Lawrence Erlbaum.

Ram, A. (1993b). Indexing, elaboration and refinement: Incremental learning of explanatory cases. *Machine Learning*, 10(3):201–248.

Ram, A. and Francis, A. (1996). Multi-plan retrieval and adaptation in an experience-based agent. In *(Leake, 1996)*, pages 167–183.

Ram, A. and Santamaría, J. C. (1993a). Continuous case-based reasoning. In *Proc. of AAAI-93 Workshop on CBR*, Washington, DC.

Ram, A. and Santamaría, J. C. (1993b). A multistrategy case-based and reinforcement learning approach to self-improving reactive control systems for autonomous robotic navigation. In *Proc. of the 2nd Int. Workshop on Machine Learning*, Harpers Ferry, WV.

Ramakrishnan, R. (1998). *Database Management Systems*. McGraw Hill.

Ricci, F. and Avesani, P. (1995). Learning a local similarity metric for case-based reasoning. In *Proceedings of ICCBR-95*, Sesimbra, Portugal. Springer-Verlag.

Rips, L. J. (1989). Similarity, typicality, and categorization. In *Similarity and Analogical Reasoning*, pages 21–59, New York. Cambridge University Press.

Rissland, E. L. and Ashley, K. D. (1986). Hypotheticals as heuristic device. In *Proc. of AAAI-86*, pages 289–297, Philadelphia, PA.

Rissland, E. L. and Ashley, K. D. (1987a). A case-based system for trade secrets law. In *Proc. of the 1st International Conference on Artificial Intelligence and Law*.

Rissland, E. L. and Ashley, K. D. (1987b). HYPO: A case-based reasoning system. In *Proc. of the 10th IJCAI*, Milan, Italy.

Rissland, E. L., Daniels, J. J., Rubinstein, Z. B., and Skalak, D. B. (1993). Case-based diagnostic analysis in a blackboard architecture. In *Proc. of AAAI-93*, Washington, DC.

Rittri, M. (1991). Using types as search keys in functional libraries. *Journal of Functional Programming*, 1(1).

Ross, B. H. (1989). Remindings in learning and instruction. In *Similarity and Analogical Reasoning*, pages 438–469, New York. Cambridge University Press.

Rubin, S. H. (1992). Case-based learning - A new paradigm for automated knowledge acquisition. *ISA Transactions*, 31(2):181–209.

Ruspini, E. H. (1990). Similarity-based interpretations of fuzzy-logic concepts. In *Proc. of the Int. Conference on Fuzzy Logic and Neural Networks*, pages 735–738, Iizuka, Japan.

Ruspini, E. H. (1991). Approximate reasoning: Past, present, future. *Information Sciences*, 57-58:297–317.

Russell, S. (1986). Quantitative analysis of analogy by similarity. In *Proc. of AAAI-86*, pages 284–288, Philadelphia, PA.

Russell, S. (1989). *The Use of Knowledge in Analogy and Induction*. Morgan Kaufmann, San Mateo, CA.

Saaty, T. L. (1996). Thoughts on decision making. *OR/MS Today*, (4):8–9.

Salzberg, S. (1991). Distance metrics for instance-based learning. In Z.W.Raś and Zemankova, M., editors, *Proc. of the 6^{th} International Symposium on Methodologies for Intelligent Systems*, pages 399–408, Charlotte, N.C.

Schank, R. C. (1982). *Dynamic Memory: A Theory of Learning in Computers and People*. Cambridge University Press, New York, NY.

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

Schank, R. C., Brand, M., Burke, R., Domeshek, E., Edelson, D. J., Ferguson, W., Freed, M., Jona, M., Krulwich, B., Ohmaye, E., Osgood, R., and Pryor, L. (1990). Towards a general content theory of indices. In *AAAI Spring Symposium on Case-Based Reasoning*, pages 36–40.

Schank, R. C. and Leake, D. B. (1989). Creativity and learning in a case-based explainer. *Artificial Intelligence*, 40:353–385.

Schuurmans, D. and Greiner, R. (1995). Learning to classify incomplete examples. In *Computational Learning Theory and Natural Learning Systems: Addressing Real World Tasks*, New York, N.Y. ACM.

Schwanke, R. W. (1991). An intelligent tool for re-engineering software modularity. In *Proc. 14th Conference on Software Engineering*, pages 83–92, Austin, TX.

Scott, P. and Sage, K. (1992). Why generalize? Hybrid representations and instance-based learning. In *Proc. of the 10th ECAI*, pages 484–486, Vienna, Austria.

Segre, A. and Elkan, C. (1994). A high-performance explanation-based learning algorithm. *Artificial Intelligence*, 69(1-2):1–50.

Segre, A., Elkan, C., and Russell, A. (1991). A critical look at experimental evaluations of EBL. *Machine Learning*, 6(2):183–195.

Seifert, C. M. (1994). The role of goals in retrieving analogical cases. In Barnden, J. A. and Holyoak, K. J., editors, *Advances in connectionist and neural computation theory. Analogy, metaphor, and reminding*, volume 3, pages 95–125, Norwood, NJ. Ablex.

Shavlik, J., Mooney, R., and Towell, G. (1991). Symbolic and neural learning algorithms - an experimental comparison. *Machine Learning*, 6(2):111–143.

Shimony, S. (1993). The role of relevance in explanation .1. Irrelevance as statistical independence. *International Journal of Approximate Reasoning*, 8(4):281–324.

Shinn, H. S. (1988). Abstractional analogy: A model of analogical reasoning. In *Proc. of 1st Workshop on CBR*, pages 370–387, Clearwater Beach, FL.

Simmons, R. G. (1988). A theory of debugging. In *Proc. of 1st Workshop on CBR*, pages 388–401.

Simon, H. (1983). Why should machines learn? In Michalski, R., Carbonell, J., and Mitchell, T., editors, *Machine learning. An artificial intelligence approach*, volume 1, pages 25–38, San Mateo, CA. Morgan Kaufmann.

Simoudis, E. (1992). Using case-based retrieval for customer technical support. *IEEE Expert*, 7(5):7–12.

Simoudis, E. and Miller, J. (1990). Validated retrieval in case-based reasoning. In *Proc. of AAAI-90*, pages 310–315, Boston, MA.

Simpson, R. (1985). *A computer model of case-based reasoning in problem solving: An investigation in the domain of dispute mediation.* PhD thesis, School of Information and Computer Science, Georgia Institute of Technology, Atlanta, GA.

Skalak, D. B. (1992). Representing cases as knowledge sources that apply local similarity metrics. In *Proc. of the 14th Annual Conference of the Cognitive Science Society*, pages 325–330, Bloomington, IN.

Skalak, D. B. (1993). Using genetic algorithm to learn prototypes for case retrieval and classification. In *Proc. of AAAI-93 Workshop on CBR*, Washington, DC.

Smee, A. (1851). *The Process of Thought Adapted to Words and Language Together with a Description of the Relational and Differential Machines.* Longman, London, UK.

Smith, L. B. (1989). From global similarities to kinds of similarities: The construction of dimensions in development. In *Similarity and Analogical Reasoning*, pages 146–178, New York. Cambridge University Press.

Smith, L. B. and Heise, D. (1990). Perceptual similarity and conceptual structure. Technical Report 30, Indiana University, Department of Psychology, Bloomington, IN.

Smyth, B. and Cunningham, P. (1992). Déjà Vu: A hierarchical case-based reasoning system for software design. In *Proc. of the 10th ECAI*, pages 587–589, Vienna, Austria.

Smyth, B. and Keane, M. (1996). Designing a la Deja Vu: Reducing the adaptation overload. In *(Leake, 1996)*, pages 151–165.

Smyth, B. and Keane, M. T. (1995). Remembering to forget: A competence-preserving case deletion policy for case-based reasoning systems. In *Proc. of the 14th IJCAI*, pages 377–382, Montreal, Quebec.

Spanoudakis, G. and Constantopoulos, P. (1994). Similarity for analogical software reuse: A computational model. In *ECAI-94*, pages 18–22.

Spiro, R. J., Feltovich, P. J., Coulson, R. L., and Anderson, D. K. (1989). Multiple analogies for complex concepts: Antidotes for analogy-induced misconception in advanced knowledge acquisition. In *Similarity and Analogical Reasoning*, pages 498–531, New York. Cambridge University Press.

Stanfill, C. and Waltz, D. L. (1986). Toward memory-based reasoning. *Communications of the ACM*, 29(12):1213–1228.

Stepp, R. E. and Michalski, R. S. (1986). Conceptual clustering: Inventing goal-oriented classifications of structured objects. In Michalski, R. S., Carbonell, J. G., and Mitchell, T. M., editors, *Machine Learning: An Artificial Intelligence Approach*, pages 471–498, Los Altos, CA. Morgan Kaufmann.

Subramanian, D. and Genesereth, M. R. (1987). The relevance of irrelevance. In *Proc. of the 10th IJCAI*, pages 416–422, Milan, Italy.

Sun, R. (1995). Robust reasoning: Integrating rule-based and similarity-based reasoning. *Journal of Artificial Intelligence*, 75(2):241–295.

Sutton, R., editor (1992). *Machine Learning*, volume 8. A special issue on reinforcement learning.

Suzuki, H., Ohnishi, H., and Shigemasu, K. (1992). Goal-directed processes in similarity judgement. In *Proc. of the 14th Annual Conference of the Cognitive Science Society*, pages 343–348, Bloomington, IN.

Sycara, K. P. (1987). *Resolving adversarial conflicts: An approach to integrating case-based and analytic methods.* PhD thesis, School of Information and Computer Science, Georgia Institute of Technology, Atlanta, GA.

Tanaka, T., Hattori, M., and Sueda, N. (1992). Use of multiple cases in case-based design. In *Proc. of the 8th IEEE Computer Society Conference on Artificial Intelligence Applications*, pages 233–239, Monterey, CA.

Tedjini, M., Thomas, I., Benoliel, G., Gallo, F., and Minot, R. (1990). A query service for a software engineering database system. In *Proc. Fourth Symposium on Software Development Environments*, pages 238–248, Irvine.

Termsinsuwan, P., Cheng, Z. X., and Shiratori, N. (1996). A new approach to ADT specification support based on reuse of similar ADT by the application of case-based reasoning. *Information and Software Technology*, 38(9):555–568.

Thagard, P. R. and Holyoak, K. J. (1989). Why indexing is the wrong way to think about analog retrieval. In *Proc. of 2 nd Workshop on CBR*, pages 36–40, Pensacola Beach, FL.

Thagard, P. R., Holyoak, K. J., Nelson, G., and Gotchfeld, D. (1990). Analog retrieval by constraint satisfaction. *Artificial Intelligence*, 46:259–310.

Thompson, K. and Langley, P. (1989). Organization and retrieval of composite concepts. In *Proc. of 2 nd Workshop on CBR*, pages 329–333, Pensacola Beach, FL.

Thrun, S. B. and Schwartz, A. (1993). Issues in using function approximation for reinforcement learning. In *Proceedings of the 4th Connectionist Models Summer School*, Hillsdale, NJ. Lawrence Erlbaum Publisher.

Turner, R. M. (1989). *A schema-based model of adaptive problem solving.* PhD thesis, School of Information and Computer Science, Georgia Institute of Technology, Atlanta, GA. Also published as Technical Report GIT-ICS-89/42.

Turney, P. D. (1995). Cost-sensitive classification: Empirical evaluation of a hybrid genetic decision tree induction algorithm. *Journal of Artificial Intelligence Research*, 2:369–409.

Tversky, A. (1977). Features of similarity. *Psychological Review*, 84(4):327–352.

Tversky, A. and Gati, I. (1982). Similarity, separability, and the triangle inequality. *Psychological Review*, 89:123–154.

Tzafestas, S. (1995). Neural networks in robot control. *Artificial Intelligence in Industrial Decision Making, Control and Automation*, 14:327–387.

Utgoff, P. E. (1986). Shift of bias for inductive concept learning. In Michalski, R. S., Carbonell, J. G., and Mitchell, T. M., editors, *Machine Learning: An Artificial Intelligence Approach*, pages 107–148, Los Altos, CA. Morgan Kaufmann.

Veloso, M. and Stone, P. (1995). FLECS: Planning with a flexible commitment strategy. *Journal of Artificial Intelligence Research*, 3:25–52.

Veloso, M. M. and Carbonell, J. G. (1993). Derivational analogy in PRODIGY: Automating case acquisition, storage,and utilization. *Machine Learning*, 10(3):249–278.

Vila, M. A., Cubero, J. C., Medina, J. M., and Pons, O. (1996). A conceptual approach for dealing with imprecision and uncertainty in object-based data models. *International Journal of Intelligent Systems*, 11:791–806.

Vosniadou, S. (1989). Analogical reasoning as a mechanism in knowledge acquisition: A developmental perspective. In *Similarity and Analogical Reasoning*, pages 413–437, New York. Cambridge University Press.

Vosniadou, S. and Ortony, A. (1989). *Similarity and Analogical Reasoning*. Cambridge University Press, New York.

Voß, A. (1994). Similarity concepts and retrieval methods. Technical Report 13, GMD, Sankt Augustin, Germany.

Voß, A., Coulon, C.-H., Gräther, Linowski, B., Schaaf, J. W., Bartsch-Spörl, B., Börner, K., Tammer, E., Bürschke, H., and Knauff, M. (1994). Retrieval of similar layouts. In Gero, J. and Sudweeks, F., editors, *Artificial Intelligence in Design*, pages 625–640, Netherlands. Kluwer Academic Publisher.

Wallach, M. A. (1958). On psychological similarity. *Psychological Review*, 65(2):103–116.

Waltz, D. (1989). Is indexing used for retrieval? In *Proc. of 2 nd Workshop on CBR*, pages 41–44, Pensacola Beach, FL.

Wang, D., Soh, Y. C., and Cheah, C. C. (1995). Robust motion and force control of constrained manipulators by learning. *Automatica*, 31(2):257–262.

Wang, H. Q. (1996). Repositories for co-operative information systems. *Information and Software Technology*, 38(5):333–341.

Watanabe, H., Okuda, K., and Fujiwara, S. (1995). A strategy for forgetting cases by restricting memory. *IEICE Trans. on Information and Systems*, E78D(10):1324–1326.

Watanabe, S. (1985). *Pattern Recognition. Human and Mechanical.* John Wiley, New York.

Watson, I. D. (1997). *Applying Case-Based Reasoning: Techniques for Enterprise Systems.* Morgan Kaufmann Publishers, San Francisco, CA.

Weiss, M. and Zeyer, F. (1994). Redesign of local area network using similarity-based adaptation. In *Proc. of the 10th IEEE Computer Society Conference on Artificial Intelligence Applications*, pages 284–290, San Antonio, TX.

Wettschereck, D., Aha, D. W., and Mohri, T. (1995). A review and comparative evaluation of feature weighting methods for lazy learning algorithms. In *Proc. of the 1st International Conference on Case-Based Reasoning*, Portugal.

Wettschereck, D. and Dietterich, T. (1995). An experimental comparison of the nearest neighbor and nearest hyperrectangle algorithms. *Machine Learning*, 19(1):5–27.

Whitaker, L. A., Wiggins, S., and Klein, G. A. (1989). Using qualitative or multi-attribute similarity to retrieve useful cases from a case base. In *Proc. of 2 nd Workshop on CBR*, pages 345–347, Pensacola Beach, FL.

Whiteley, J. and Davis, J. (1994). A similarity-based approach to interpretation of sensor data using adaptive resonance theory. *Computers & Chemical Engineering*, 18(7):637–661.

Widmer, G. and Kubát, M. (1993). Effective learning in dynamic environments by explicit context tracking. In *European Conference on Machine Learning*, Vienna, Austria.

Wixon, D., Holtzblatt, K., and Knox, S. (1990). Contextual design: An emergent view of system design. In *Proc. of CHI-90*, pages 329–336.

Wolverton, M. and Hayes-Roth, B. (1994). Retrieving semantically distant analogies with knowledge-directed spreading activation. In *Proc. of AAAI-94*, Seattle, WA.

Yang, S.-A., Robertson, D., and Lee, J. (1993). KICS: A knowledge-intensive case-based reasoning system for building regulations and case histories. In *Proc. of the 4rd International Conference on Artificial Intelligence and Law*, Amsterdam, The Netherlands.

Zuenkov, M. A., Kulguskin, A. S., and Poletykin, A. G. (1986). Forming similarity relations in analogy-driven systems. *Automation and Remote Control*, 47(11 Part 2):1543–1551.

Appendix A

Appendix

A.1 Case Representation Examples

This section shows several case representation formalisms. Specific approaches are presented in Figures A.1-A.6. Namely, we show how cases are represented in CHEF (Hammond, 1989a), the Cardie system (Cardie, 1993a), PROTOS (Bareiss, 1988), FindMe (Hammond, Burke and Schmitt, 1996), and ESPANDA (Garben, Furnsinn and Ruschkowski, 1995).

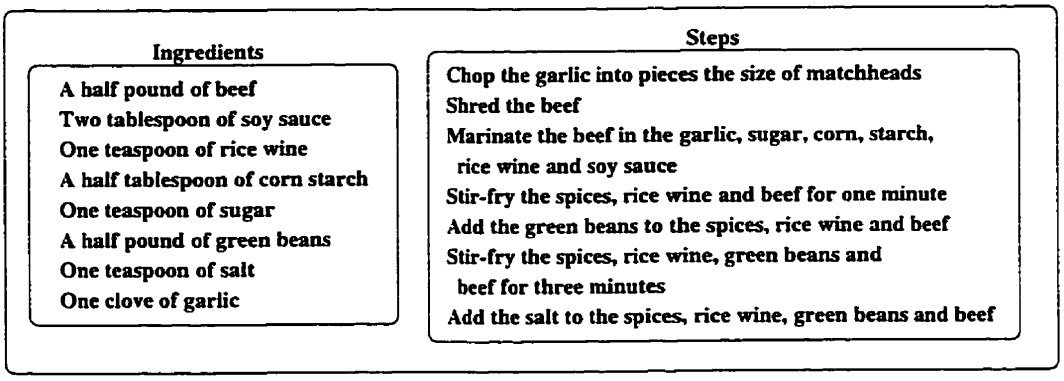


Figure A.1: *CHEF's recipe: Beef with green beans.*

A.2 Case Retrieval Examples

This section discuss a few well-known approaches to case retrieval.

JULIA (Kolodner, 1987) presents a piece of an old plan as a ballpark solution if it is attempting to derive some piece of a meal plan; it proposes a similar meal plan as a ballpark solution, if the task is to derive a meal plan.

PROTOS (Bareiss, 1988) uses a coarse evaluation function to distinguish which of the many

```
(DEFMOP M-RECIPE (M-CASE)
    (MEAT M-MEAT) (VEGE M-VEGETABLE)
    (STEPS M-PATTERN (CALC-FN ADAPT-STEPS)))

(DEFMOP I-M-BEEF-AND-GREEN-BEANS (M-RECIPE)
    (MEAT I-M-BEEF) (VEGE I-M-GREEN-BEANS)
    (STYLE I-M-STIR-FRIED)
    (STEPS M-RECIPE-STEPS
        (BONE-STEPS I-M-EMPTY-GROUP)
        (CHOP-STEPS M-STEP-GROUP
            (1 M-CHOP-STEP (OBJECT I-M-BEEF)))
        (LET-STAND-STEPS M-STEP-GROUP
            (1 M-LET-STAND-STEP
               (OBJECT M-INGRED-GROUP
                       (1 I-M-BEEF)
                       (2 I-M-SPICES))))
        (STIR-FRY-STEPS M-STEP-GROUP
            (1 M-STIR-FRY-STEP
               (OBJECT M-INGRED-GROUP
                       (1 I-M-BEEF)
                       (2 I-M-SPICES)))
            (2 M-STIR-FRY-STEP
               (OBJECT M-INGRED-GROUP
                       (1 I-M-BEEF)
                       (2 I-M-GREEN-BEANS)
                       (3 I-M-SPICES))))
        (SERVE-STEPS M-STEP-GROUP
            (1 M-SERVE-STEP
               (OBJECT M-INGRED-GROUP
                       (1 I-M-BEEF)
                       (2 I-M-GREEN-BEANS)
                       (3 I-M-SPICES))))))
```

Figure A.2: *CHEF: case representation.*

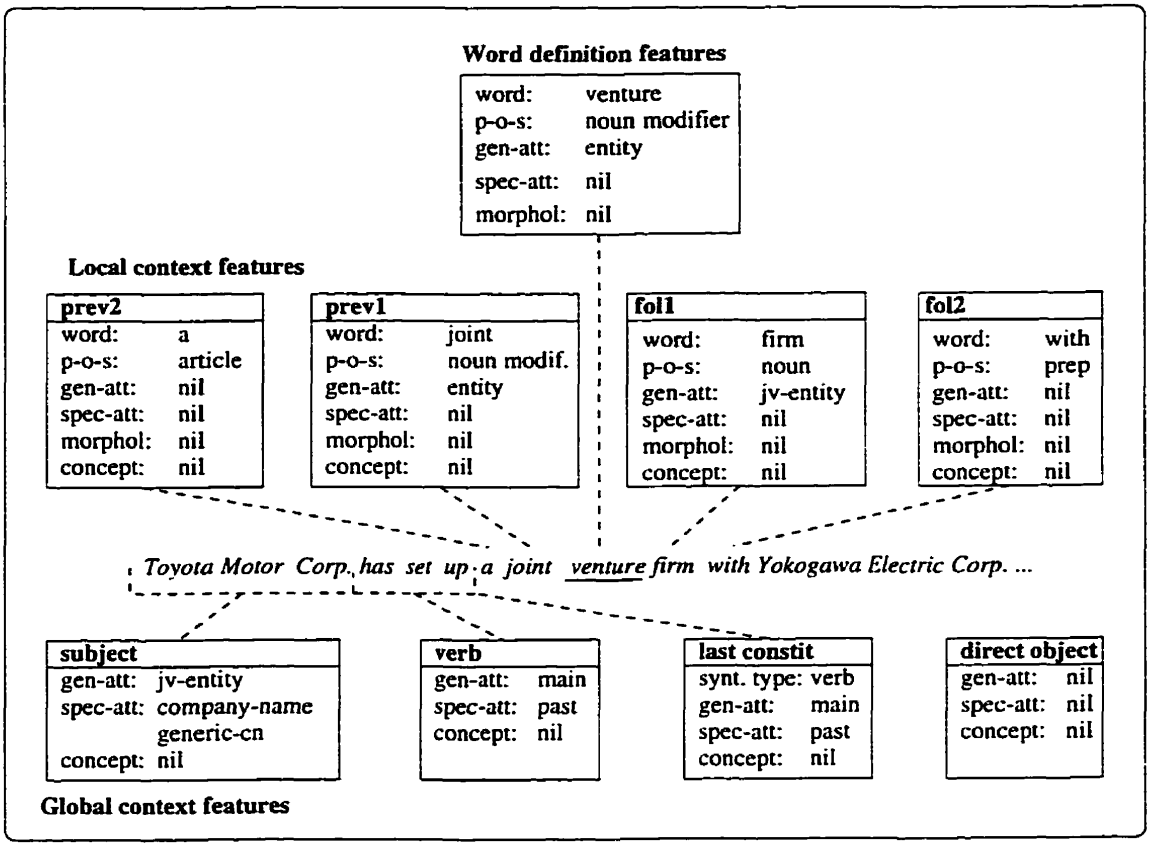


Figure A.3: *Case representation in Cardie system.*

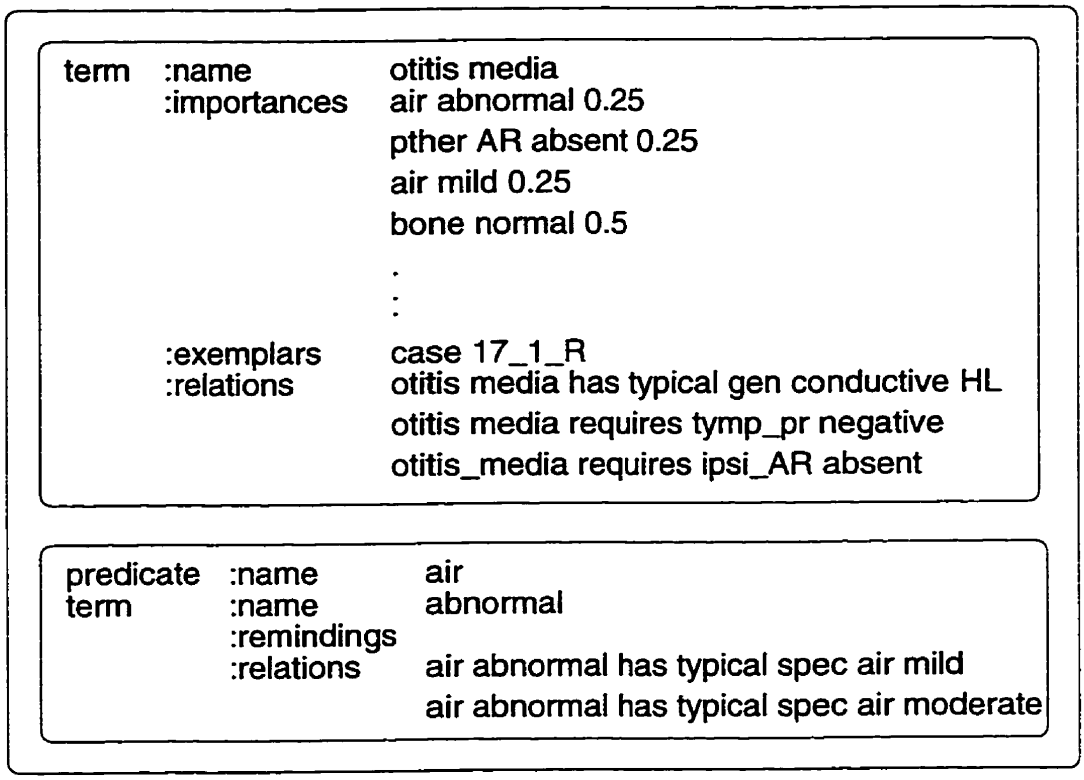


Figure A.4: *PROTOS: case representation.*

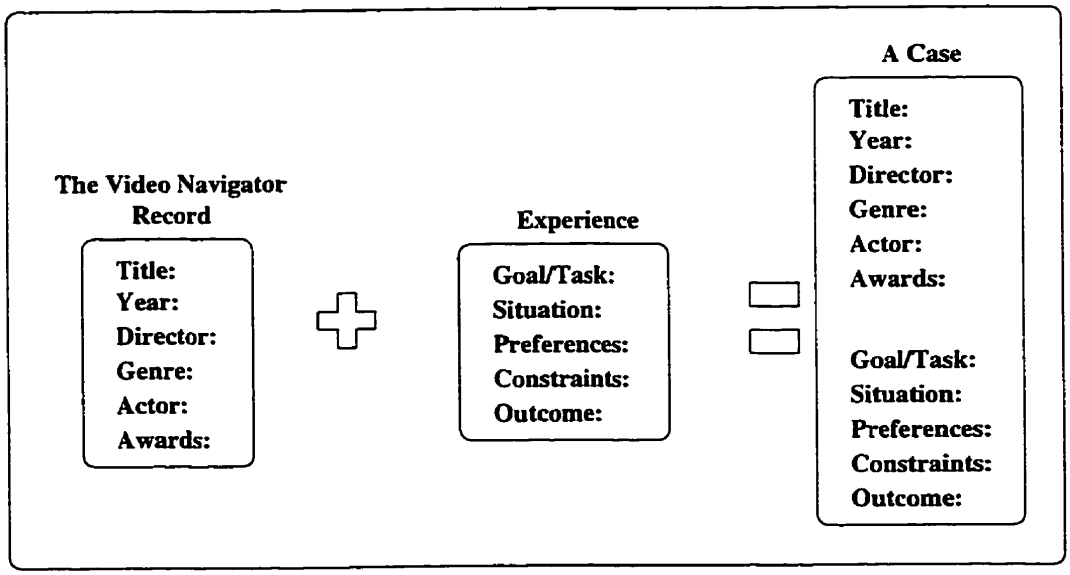


Figure A.5: *Generating a case from a database record in FindMe.*

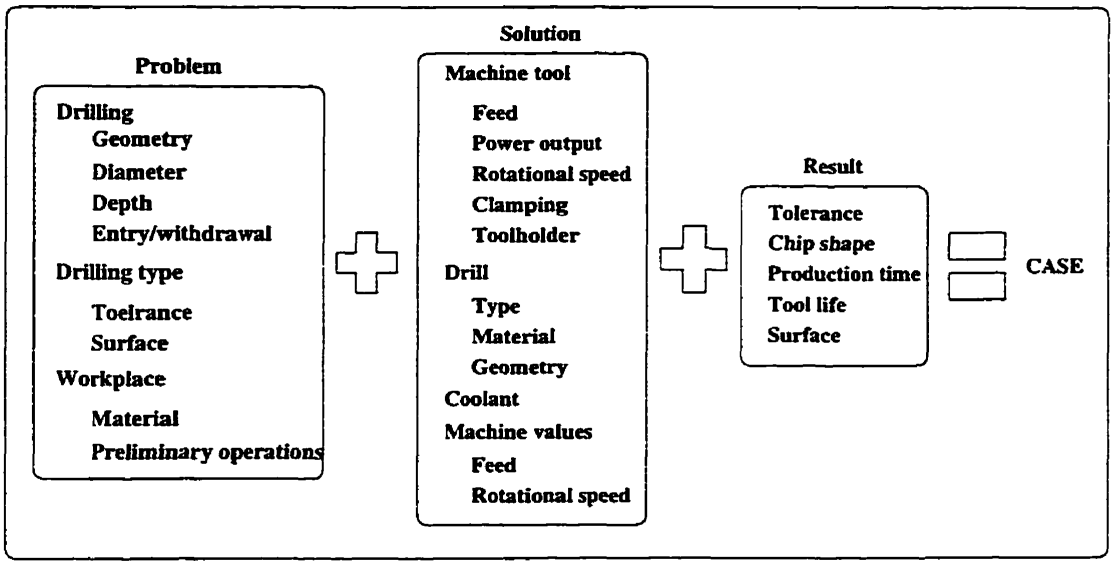


Figure A.6: *ESPANDA: case representation.*

possible interpretations proposed by the recalled cases is closest to its new case.

PARADYME (Kolodner, 1989) uses preference heuristics after retrieval to choose the most useful cases from the set of cases returned by memory access functions. Previous cases are used to provide context to the case selection process.

HYPO (Ashley, 1990) does not start to work in this step since it has necessary information already given in the form of dimensions. CLAVIER (Barletta and Hennessy, 1989; Hennessy and Hinkle, 1992) proposes several relevant cases; one case is presented as an overall layout and the other cases are used to complete the layout where required.

CBA (Case-Based Appraisal) (Gonzalez and Laureano-Ortiz, 1992) retrieves ten most similar cases which are used to compute the similarity match score. The most similar case is used as a basis for a new appraisal, while the others serve as a guideline for an adaptation of it.

The protein secondary structure prediction system (Leng, Buchanan and Nicholas, 1993) retrieves first k similar cases (protein structures) from the case base (the number of selected cases is variable and depends on the precision of the similarity metric). The most similar protein is used as a reference case. In the second step, the sequence is decomposed into pieces and the structures are analyzed.

A.3 Case Adaptation

This section discuss a few well-known examples of case adaptation. Individual applications use a set of domain-dependent adaptation rules (Hammond, 1989a; Kass, 1991). In addition to domain dependency, the rules in CHEF (Hammond, 1989a) are usable only for local changes; they cannot be used to make global changes to the whole recipe. Difficulty in obtaining reliable adaptation rules,

together with an impossibility of using these specific rules in other domains, is an obvious problem of domain-dependent adaptation.

The adaptation in ABE (Kass, 1990) is based on tweaking strategies which are used to make small changes to explanation patterns. There are three strategies available: replace slot fillers with causal equivalent, generalize over-specific explanations, and elaborate incomplete explanations. Used strategies encode general knowledge about planning, not the knowledge about the specific planning domain. Thus, ABE is less domain-dependent than CHEF.

Some systems use a domain model (Koton, 1988b; Jones, 1992). The model is used to adapt the retrieved diagnosis, taking into account differences in symptoms between retrieved and input cases (Koton, 1988b, CASEY).

In the case of BRAINSTORMER (Jones, 1992), the adaptation converts generic knowledge into specific knowledge. The model-based adaptation suffers from the same problems as a set of adaptation rules in CHEF.

Another approach is to use pieces of existing cases during adaptation (Hennessy and Hinkle, 1992; Gonzalez and Laureano-Ortiz, 1992; Tanaka, Hattori and Sueda, 1992). This is the most promising approach since it is the most general one and requires no additional domain knowledge other than cases. However, the problem of solution validation is more severe here, when compared to previous methods.

A.4 Case Base Organization

The purpose of this section is to describe knowledge classification techniques used to improve managing of large and complex case bases by partitioning a case base into a meaningful and manageable parts. Our motivation is the need for scalable systems.

Different systems use different classification methods. Porter categorizes the methods on the basis of the source of knowledge for classification (Porter, 1989). He identifies two possibilities: an explicit representation using a fixed vocabulary, and a dynamic representation which stores cases in hierarchies (because there is no predefined structure, this approach is more flexible and adaptable). An obvious advantage of the latter approach is the adaptability and flexibility which makes it suitable for large case bases and especially for inter-domain reasoning. The other categorization is on the basis of the algorithm used, namely:

- **Nearest-neighbor** retrieves cases based on a weighted sum of features in the input case that match cases in memory (Hennessy and Hinkle, 1992; Pearce et al., 1992; Gonzalez and Laureano-Ortiz, 1992; Cardie, 1993a). The method is effective if the retrieval goal is not well defined or if few cases are available. The problem with the method is that the feature weights are context dependent and thus they must be either predefined (and consequently fixed) or a

special method must be used (e.g., inductive learning) to specify them in more dynamic way (not necessarily after each case acquisition).

- The **inductive** approach (e.g., ID3 (Quinlan, 1986b), UNIMEM (Lebowitz, 1987), COBWEB (Fisher, 1987)) automatically extracts relevant features for retrieval or creates hierarchies of cases by comparing the classification and discrimination abilities (the ability to characterize and distinguish a particular case from the others) of individual features (Goodman, 1989; Simoudis, 1992). This method is applicable only if sufficient number of cases is available and the goal of retrieval is well-defined. However, the learning process is quite time consuming. This is the reason why in some applications the classification is done only as a batch process. In some implementations, only surface features (easy to observe) are classified using the inductive method while deep features (difficult to observe) are classified by a human expert.

- **A discrimination net** uses both previous approaches (Kolodner, 1983; Bradtke and Lehnert, 1988; Hammond, 1989a; Cardie, 1993a; Rissland et al., 1993).

- An **explanation-based** approach uses past experience to alter the classification (DeJong and Mooney, 1986; Barletta and Mark, 1988; Ram, 1993b). The main advantage of the method is its applicability to support the case classification. This process helps to incrementally refine the system's understanding of the case. The EBI system (Explanation-Based Indexing) (Barletta and Mark, 1988) can use any automatically selected feature for a classification; however, the system may select features which are hard to observe (or compute) or which does not provide sufficient discrimination (e.g., many cases would be retrieved using this feature). The AQUA system (Asking Questions and Understanding Answers) (Ram, 1993b; Ram, 1993a) overcomes this drawback by selecting only easy-to-observe or easy-to-compute features. But even this approach has some drawbacks: only predefined types of classifications may be learned and in some cases, a classification may be created for a one-time-only event.

- A **knowledge-based** approach uses existing knowledge of each case in the case library to determine the importance of individual features for retrieving each case (Lauzon and Rose, 1994; Kitano, Shibata and Shimazu, 1993). Even though all approaches up to the present rely on domain expertise, this approach is the most interesting due to its adaptability and flexibility, which makes it suitable for inter-domain problem-solving.

The problem of organizing cases so that the retrieval of relevant experience is efficient is one of the most important issues in case-based reasoning. An attempt has been made to unify different approaches to the indexing problem (Schank et al., 1990). The authors present ideas about the representational issues required to support efficient retrieval. Domeshek presented additional theoretical work on this problem (Domeshek, 1992). However, in both cases only certain domains (story

understanding) were considered and no attempts were made to support other domains.

On the basis of presented differences in approaches to knowledge retrieval we define the following categorization:

- the algorithm for knowledge classification either uses a certain (predefined) class of features which are used in accessing cases (Kolodner, 1983; Birnbaum and Collins, 1989; Hunter, 1989; Owens, 1989; Hammond, 1989a) or is based on the representation hierarchy (Porter et al., 1988; Lauzon and Rose, 1994);
- knowledge can be stored as is or a generalization hierarchy may be created (Shinn, 1988; Branting, 1989; Thompson and Langley, 1989; Collins and Birnbaum, 1990; Fertig and Gelertner, 1991; Jones, 1991; Yang, Robertson and Lee, 1993; Ram, 1993b);
- the algorithm can depend heavily on a special computer architecture (Stanfill and Waltz, 1986; Kolodner, 1989; Waltz, 1989; Rubin, 1992), or it can be implemented on a standard computer;
- the classification algorithm assumes large, real-world case bases (Kitano, Shibata and Shimazu, 1993) or it can assume only small ones;
- in addition to functionality, the algorithm may or may not be psychologically plausible.

The process of classification is crucial for the system's performance. If the system uses a predefined set of features for a classification, then its problem-solving abilities are constrained since the system becomes a single context system (Porter, 1989). Several strategies for solving the problem of predefined indexing vocabularies were identified, namely:

1. All features are used for classification. This naive approach suggests an extension that probably not all features are equally useful. Thus, next approaches try to (somehow) select appropriate features for indexing.
2. Only less computationally expensive features are considered for classification. The problem here is that using these features during retrieval is efficient, but it may result in less than perfect retrieval.
3. Use only features given in the initial description of the problem situation ("low-level" features). This approach is simple, but may not be useful in complex domains.
4. More abstract properties of the problem description are generated and only these are used for classification. The main problem of this approach is with derivation of more abstract features, because deep knowledge may be required but unavailable in a particular application.
5. A combination of 3 and 4. This approach can overcome the drawbacks of both approaches.

Given the presented approaches we can make either the initial process (case acquisition and case base construction) more difficult or the reasoning process slower. If the initial knowledge is shallow, then the knowledge acquisition is going to be simpler than in the case when the initial knowledge has also deep structures.

Another important issue is that regardless of where the most work is done, the indexing vocabulary construction is a difficult problem; thus, there is a necessity to employ machine learning approaches to help in this process. Another reason for this is that static classification is in many domains useless since it does not take into account the task and context that may change over time (Suzuki, Ohnishi and Shigemasu, 1992; Berman and Hafner, 1993; Kitano, Shibata and Shimazu, 1993; Lauzon and Rose, 1994).

Indexing

The basic assumption of the CBR system is that all relevant cases will be retrieved efficiently. In order to guarantee efficient retrieval, case storage requires special techniques to be applied. Several general approaches have been proposed for this problem, namely:

1. Classification of all cases into a hierarchy, grouping relevant cases together (this process is usually referred to as the *indexing problem*.
2. Implementation of a CBR system on a parallel computer without using case classification (Stanfill and Waltz, 1986; Waltz, 1989; Rubin, 1992).
3. Classification is supported by parallel implementation (Kolodner, 1989).
4. Neither classification nor the parallel implementation is used (Thagard et al., 1990; Hennessy and Hinkle, 1992; Gentner and Forbus, 1991). However, this approach is only usable for small case bases.

A.5 Relation of CBR to Other Areas in AI

A.5.1 Learning

A CBR system is a problem-solving system with inherently incorporated learning. Several techniques are used at various stages. **Accumulation of new cases** constitutes learning by remembering, knowledge acquisition or rote learning. New cases give the reasoner more context for solving problems or evaluating situations. Learning can also help to estimate which cases are worth storing and which should not be included in the case base. Learning algorithms can also help in converting existing databases into case bases. **Automatic case generation** can be supported by analogical reasoning, explanation-based learning and by using genetic algorithms (construction of hypothetical cases).

(Re)structuring the knowledge in a case base involves classification and changes of hierarchy of knowledge base concepts. New classification enables fine-tuning a system's recall strategy, so it remembers cases at more appropriate times. Hierarchical changes affect system's deep knowledge. **Updating the existing knowledge** is a form of reinforcement learning.

Generalization of already possessed knowledge involves creating more general concepts and case modification (construction of generic cases, rule generation, domain theories revision and development). When several cases are represented as similar to each other, and all predict the same solution, the reasoner can form a useful generalization. On the one hand, there are systems which use only instances of cases; on the other hand, there are systems which use both instances of cases and their generalizations. There are also systems which use the generalization process as a way of evolving useful rules from cases (Boström and Idestam-Almquist, 1989). Rules could be used when they match cases exactly, while cases would be used when rules were not immediately applicable. An alternative approach is when a system uses only abstract cases (Jones, 1992).

Usually, the learning in CBR systems is opportunistic, which means that it is invoked only when some opportunities are recognized and it is not planned in advance. An example of such learning is failure-driven learning where goals and situations that are likely to cause problems in the future are identified (Hammond, 1989a; Hunter, 1989; Hammond et al., 1993).

A.5.2 Analogy

Analogical problem solving has been used in CBR systems in various ways. CYRUS (Kolodner, 1983) is the first computer implementation of many of the themes expressed in Schank's work (Schank, 1982). It is a story understanding program that uses analogical problem solving if it cannot find a direct answer to a query. Another example of analogical problem solving using a case-based approach is SWALE (Kass and Leake, 1988), a case-based creative explainer. When SWALE cannot explain an anomalous event, it starts to look for other explanations in other contexts. The analogical reasoner in the planning system BRAINSTORMER (Jones, 1989) is similar to TWEAKER (a descendant of the SWALE system, which focuses on the tasks of explanation application and adaptation (Kass and Leake, 1988)); it is a failure-driven process and is invoked to answer requests from the planner.

Analogical problem solving has been studied as a psychological model. In SME (Structure Mapping Engine) structural and semantic constraints are used to generate all possible matches between two domains (Falkenhainer, Forbus and Gentner, 1989; Gentner, 1983). Later, the resulting set is processed using structural criteria and the best match set is identified. In ARCS (Analog Retrieval by Constraint Satisfaction) analogs are retrieved using multiple constraints (Thagard et al., 1990). In MAC/FAC (Gentner and Forbus, 1991), a two-stage retrieval is used as in ARCS. First, analogs are retrieved on the basis of superficial similarities. Second, a retrieved set of analogs is refined on the basis of structural similarities (a second stage is similar to the SME approach).

Even though there is some overlap between analogical and case-based reasoning, each has some distinctive features. Case-based systems aim to find useful situations while analogical systems tend to find similar situations. Systems using analogy are usually inter-domain, whereas CBR systems are intra-domain. Most CBR systems retrieve specific cases from a case base as opposed to generalized concepts or domain principles. An exact match is considered to be a perfect case in CBR; however, is not considered as an analogical relationship. Case-based systems usually pay less attention to the overall structural consistency of the retrieved case and the new problem as long as the new case is "sufficiently" similar in a way that it makes it useful to the process of solving a given problem. Models of analogy have typically been more concerned with overall systematicity. Case-based systems are pragmatic, problem solving, and task oriented. whereas analogical systems have broader scope of reasoning in learning and generalization.

According to psychological studies on analogy, both goal-based features and surface features are important for retrieval of related cases from a case base. The decision which of the two extreme approaches is more important is domain dependent. It was pointed out, however that what is a surface feature in one domain can easily be a structural feature in another (Keane, 1988; Vosniadou and Ortony, 1989). An interesting issue is to examine which features are more important in a particular domain and to adapt the system accordingly. It is because the value of case features affect not only the retrieval stage of analogy, but also the transfer stage.

When a similar case is identified in memory it could be unnecessarily complex for a given task, or it could be incomplete. So, it is suggested that a more sophisticated process of reconstruction should be used before retrieved information can be used in a transfer. Reminding occurs only when cases are "instructed" to be remembered. Thus, analogical reminding is not a deterministic procedure that can be evoked without strategic effort. It is also recognized that cognitive goals form a context within which retrieval operates, and the nature of this context determines what remindings occur.

Experimental tests show that deeper understanding and encoding aids analogical transfer (Keane, 1988; Vosniadou and Ortony, 1989). It is also believed that a great deal of inference is required to fully understand an example containing abstract relations as well as content features. Building an initial representation that contains both the abstract and content features is critical for any later analogical use based upon them. There is a problem that many times the system will not scale up properly for larger case bases. It is necessary that the system stores representative cases, since the similarity and thus reminding is context sensitive. It should also be guaranteed that the search space of exemplars includes many kinds of overlapping knowledge and features. The next question is when to use cases to reason. Most of the systems (unless with hybrid representation) use case-based reasoning for all new problems. But psychological tests seem to support an idea that cases are not used so often and that problem solving relies on generalized principles most of the time.

It is believed that CBR systems have the strong pragmatic, problem-solving task orientation,

as opposed to the broader role of reasoning by analogy in learning and generalization (Kolodner and Simpson, 1989). Many CBR systems to date rely on pre-existing generalizations, domain and system-specific procedures to determine what might be useful in a new case. However, it is also believed that CBR should be useful in other kinds of complex tasks that are not exclusively problem solving. This brings in an interesting issue: systems usually use AI models which allow them to work only in a particular domain because the role of the process is implicitly encoded in the model. the memory base and representational format. The issues to be addressed by CBR systems are analogical transfer between domains and abstract reminding.

A.6 Application Areas of Case-Based Systems

A.6.1 Case-Based Design

In case-based design, problems are defined as a set of constraints; the problem solver is required to provide a concrete artifact that solves the constraint problem (Kolodner, 1992). The constraints can either *under-specify* the problem, i.e., there are many possible solutions or *over-constrain* the problem, i.e., there is no solution if all constraints are fulfilled. In solving either of these cases, the design specifications must be re-specified while solving the problem.

CBR systems are especially useful for the problems that are hard to solve both in one chunk and in a decomposed form because of the strong interaction between individual pieces. Solving the problem by adapting an old solution helps the problem solver to avoid dealing with many constraints, and keeps it from having to break the problem into pieces needing recomposition. These features are especially suited for a design task.

Another interesting contribution of a CBR paradigm for design is pointing out problems with a proposed solution. In almost all design problems, more than one case is necessary to solve the problem. The reason for that is the complexity of problems and the fact that design problems tend to be large. Within the CBR approach, decomposition and recomposition are avoided, as are large constraint satisfaction and relaxation problems.

Next we describe some of the design-oriented CBR systems. JULIA (Kolodner, 1987; Hinrichs, 1988; Hinrichs, 1989) plans meals. The emphasis is put on the final product, which has to meet certain constraints rather than on individual steps towards the main goal, as it is in planning. Cases are organized according to their features in a discrimination net. CYCLOPS (Navinchandra, 1988) is applied for landscape design. KRITIK (Goel, 1989; Goel and Chandrasekaran, 1989) is a combination of case-based and model-based reasoning for design of small mechanical assemblies. Cases are used to propose solutions; the model is used to verify these solutions, to point out where adaptation is needed, and to suggest specific adaptation.

CLAVIER (Barletta and Hennessy, 1989) is being used to lay out pieces made of composite

materials in an oven to bake. There is no complete causal model of what works and why; however, the system's goal is to avoid placing pieces in the wrong places. Almost always, several cases are used to do the design. One case provides an overall layout, which is adapted appropriately, the others are used to fill in holes in the layout. This system is distinct from others because it is used on a daily basis in Lockheed Inc. since September 1990.

PERSUADER (Sycara, 1987) designs a contract for labor management disputes by adapting similar contracts. If no possible contract can be retrieved, the system generates a new contract from scratch. The emphasize is on meeting different constraints simultaneously. XBE (Pankakoski et al., 1991) is applied to manufacturing systems design. The system allows the user to represent and use different types of knowledge; it also uses hypothetical cases (similarly to the HYPO system (Ashley, 1990)). ARCHIE (Pearce et al., 1992) is used to help architects in the high-level task of conceptual design. The cases are used both for proposing and critiquing. A case consists of design goals, design plans, outcomes, and lessons to be learned. DéjàVu (Smyth and Cunningham, 1992) is a hierarchical CBR system for the development of plant control software for controlling autonomous vehicles in loading and unloading metal coils in a steel milling process.

A.6.2 Case-Based Planning

Planning is the process of coming up with a sequence of steps or a schedule for achieving a particular state of the world. There are several problems that must be dealt with in planning, namely a problem of protections and preconditions. The first one relates to ensuring that plans are correctly sequenced and that individual steps are properly projected into the future. The second one relates to assuring that all preconditions of individual steps are fulfilled before their scheduling.

In addition to the problems mentioned above, even more complex problems can arise if the number of goals competing for achievement at any time is quite high. In general, there are two possible ways to tackle this problem. If each goal is achieved independently of the others, then planning and execution time are at least the sum of achieving each goal (because of interactions, planning and execution can be even more complex). This requires decomposition and recomposition of plans. The other possibility is to achieve several goals simultaneously, which solves the goals in conjunction but it makes the problem significantly harder.

Case-based planning and case-based design systems are quite similar to each other. Both types of systems are useful for the problems that are hard to solve in one chunk, but cannot be easily decomposed, and solved separately, because of the strong interaction between individual pieces. Here, individual steps imply different constraints, which have to be satisfied.

Case-based planning uses previously successful plans, classified according to the conjunction of goals they achieve. Case-based planning inherently diminishes the problem of combining planning and execution. Some of the well-known planning systems include CHEF (Hammond, 1989a), which

works in the domain of Chinese recipes. CHEF addresses the problem of anticipating problems before execution time by learning from its problematic experiences. PLEXUS (Alterman, 1988) is a program to demonstrate the skills required to ride a subway. PLEXUS is able to do execution time repairs by adapting and substituting semantically similar steps for those that have failed. TRUCKER (Hammond, 1989b; Hammond et al., 1993) keeps track of pending goals and takes advantage of opportunities that arise and allow it to achieve these goals earlier than expected. MEDIC (Turner, 1989) is a diagnosis program which is able to reuse previous plans for diagnosis; it is flexible enough in its reuse to be able to follow up on unexpected turns of events. MEDIC also compares the importance of goals and decides which plan to use to achieve the most important goal. Because of its emphasis on planning, the system is considered to be a case-based planning system, rather than case-based diagnostic system. CSI BATTLE PLANNER (Goodman, 1989) is an example where cases are used to criticize and repair plans before they are executed. SPA (Hanks and Weld, 1992) is a domain-independent case-based planning system. A plan is represented as a set of steps, various links, and constraints describing interrelationships among the steps.

A.6.3 Case-Based Diagnosis

In case-based diagnosis, a solver is given a set of symptoms and is asked to explain them. When only a small number of possible explanations exist, diagnosis is similar to the classification problem. If the space of possible explanations is significant, one can view diagnosis as the problem of creating an explanation. Similarly to the comparison of design and planning CBR systems, case-based diagnosis system can rely on planning and thus can be classified as a planning system (Turner, 1989). However, there are also diagnosis systems, which are model-based (Koton, 1988b; Goel, 1989; Goel and Chandrasekaran, 1989; Birnbaum et al., 1990; Féret and Glasgow, 1993). In these cases, categorization is easier since the distinctive features are easily perceivable.

SHRINK (Kolodner and Kolodner, 1987) is an early CBR system, designed to be a psychiatric diagnostician. Here, the diagnosis from a previous case is used to generate a hypothesis about the diagnosis in the new case. CASEY (Koton, 1988b) is a system applied for diagnosis of heart problems; it adapts the diagnosis of previous heart patients to new patients. CASEY is built on top of an existing model-based diagnostic program. Because adaptation is based on a valid causal model, its diagnoses are as accurate as those made from scratch based on the same causal model and they also tend to be more time efficient. PROTOS (Bareiss, 1989a), a hearing disorders diagnostic system, is designed to ensure that cases are used in pointing the way out of previously experienced reasoning predicaments. In its application domain, many of the diagnoses manifest themselves in similar ways, and only subtle differences differentiate them. By using feedback from the user, PROTOS learns these subtle differences. EAD (Explanation-Aided Diagnosis) (Féret and Glasgow, 1993) is a model-based diagnostic system with a CBR module, applied for diagnosis of complex devices. The system

uses hierarchical knowledge representation and the model-based reasoner is used to identify relevant experience. The case-based module critiques the results obtained by the model-based reasoner, and also helps to consider unexplored alternative hypotheses.

A.6.4 Case-Based Explanation

Explanation is introduced to the CBR terminology in the connection with the plan or solution failure. The system may try to construct an explanation of the reason for the failure in order that the plan or solution may be repaired and the case base modified to avoid such a failure in the future.

Explaining anomalies is prevalent in all of our problem solving and understanding activities. Explanation has been called the credit assignment problem and the blame assignment problem, depending whether it explains a success or a failure.

A case-based explanation (Ram, 1993b; Ram, 1993a; Schank, 1986; Schank and Leake, 1989) explains a phenomenon by remembering a similar phenomenon, borrowing its explanation, and adapting it to fit. In this context, CBR fits very naturally and will improve the explanation process. A case-based explanation requires mechanisms for retrieving similar cases, adapting them to fit current needs and validating the solution that can decide if a proposed explanation has any merit.

In a different sense, cases can be used to explain other reasoning. For example a case-based explanation can be used to explain solutions generated by a genetic algorithm (Louis, McGraw and Wyckoff, 1992). This task is quite difficult because the solutions cannot be easily explained. An explanation is later used to tune the genetic algorithm. Thus, case-based explanation helps in developing an adaptive genetic algorithm.

A.6.5 Case-Based Classification

The classification task involves associating instances with particular classes: on the basis of the object description, the system determines whether a given object belongs to a specified class. For the purpose of this paper, it is assumed that the categories are known from the domain theory (Stepp and Michalski, 1986). In general, this process involves generating a set of categories and classifying given cases into the created categories.

Various techniques have been used for the classification task, including neural networks (Shavlik, Mooney and Towell, 1991), genetic algorithms (Frey and Slate, 1991), inductive and instance-based learning (Aha, Kibler and Albert, 1991; Kononenko and Bratko, 1991; Shavlik, Mooney and Towell, 1991; Turney, 1995; Schuurmans and Greiner, 1995) and CBR (Porter, Bareiss and Holte, 1990; Aha and Bankert, 1995; Griffiths and Bridge, 1995). Individual approaches are compared to each other on the basis of the method they deploy, accuracy they can achieve and on the complexity of the algorithm used. Most of the systems extract classification rules from training examples during a learning process. Then, they use these rules for classification of unseen instances. Case-based

systems, however, store whole cases during the learning process and use the similarity between a given problem and a case in a case base to determine a proper class for a problem.

Evaluating classifiers' performance is not a straightforward process. Individual systems are usually tested on different problem domains and because of differences in domain complexities obtained performance measures cannot be compared directly. While the performance is highly domain dependent, it is possible to derive evaluation techniques which allow for a meaningful performance comparison of different algorithms (Aha, 1992b; Kononenko and Bratko, 1991).

In a classification system, there is usually a fixed set of classes (or categories) into which given objects are to be classified. Classification systems are trained first, i.e., already-classified objects are presented to them. The system induces knowledge from these examples - e.g., neural networks change connection weights, decision-tree algorithms generate rules and CBR systems remember individual instances. The goal is to learn to correctly classify all the examples. Obviously, the system must be able to generalize from these examples, so even unseen examples would be classified correctly with satisfactory accuracy.

Generally, objects are represented as a collection of properties - attributes or features. In real-world domains, objects may or may not be represented properly and/or error-free. Some attribute values might be missing or there may be irrelevant attributes present and relevant attributes missing. Such domains are called imperfect.

In a conventional data analysis, objects are clustered into classes based on a distance (similarity) measure (Stepp and Michalski, 1986). The similarity of two objects is represented as a number - the value of a similarity function applied to symbolic descriptions of objects. Thus, the similarity measure is context free. Note that such methods fail to capture the properties of a cluster as a whole and are not derivable from properties of individual entities (Stepp and Michalski, 1986).

Supervised learning algorithms for classification can be described as follows:

> Given a sequence of training examples $\langle \bar{x}_1, c_1 \rangle, \ldots, \langle \bar{x}_n, c_n \rangle$, where
> $\langle \bar{x}_i, c_i \rangle$ is a pair consisting of a object description $\bar{x}_i$ and a proper classification c_i, the classification system must learn the mapping $classify : X \rightarrow C$, where X is the space of objects descriptions and C is the space of possible classes.

Case-based classification classifies instances based on their similarity to stored cases:

> A new problem (a case $\mathcal{C}_p$) is classified on the basis of k close matches $(\mathcal{C}_1, \ldots, \mathcal{C}_k)$ retrieved from a case base $(\Delta = \{\mathcal{C}_1, \ldots, \mathcal{C}_m\})$.

Traditional approaches to classification involve generating a set of rules based on induction from training examples. The requirement for such algorithms is that created rules correctly classify known (i.e., training) examples and perform well on unseen (i.e., test) examples (Quinlan, 1986a).

ID3 (Quinlan, 1986b) is a classification system based on a decision-tree algorithm. Based on the

induction from training examples, ID3 generates a decision tree – a classification rule that examines the values of some attributes of an object in order to assign it into a proper class.

AC^3 (Stepp and Michalski, 1986) is a classification algorithm based on attribute-based conjunctive conceptual clustering – a way of grouping objects into conceptually simple classes. Using this method, a set of objects forms a class only if it can be described by a conjunctive concept involving relations on object attributes.

AUTOCLASS II (Cheeseman et al., 1988) is an induction algorithm used to discover classes from databases on the basis of Bayesian statistical techniques. The system determines the number of classes, their probabilistic descriptions and the probability that each object is a member of a given class. This allows for making each and every attribute potentially significant as well as assigning objects to different classes. The system also allows for identifying hierarchies of attributes, selecting common attributes and distinguishing attributes between classes.

The STABB system (Utgoff, 1986) uses induction and shifting bias for concept learning. The author of the system also identifies several factors as contributors to bias.

IB1 (Aha, 1992b; Aha and Salzberg, 1994) is a nearest-neighbor, instance-based learning system used for classification and control tasks. Given a problem description, IB1 provides a solution to it using the k-nearest neighbors. Since all attributes are used during retrieval, the system's performance decreases with the number of irrelevant attributes. Performance degradation of IB1 with the number of irrelevant attributes can be solved either by data pre-processing (removing all irrelevant attributes) or by the use of an intelligent, selective partial matching algorithm. The latter approach is similar to m-of-n concepts in machine learning (Ortega, 1995), i.e., the system considers only $m < n$ attributes during retrieval. There are various methods used to decide m and Ortega presents an evaluation of system performance for various m (Ortega, 1995) . In addition to deciding the size of m, the approach described in this paper allows for specifying which attributes to exclude and for posing additional constraints on attribute values, if desired.

CB1 (Griffiths and Bridge, 1995) is a PAC (Probably Approximately Correct) learning, case-based classification algorithm designed for general purpose learning. Even though it is relatively inefficient, it is an interesting step towards a computational learning theory for CBR systems.

PROTOS (Porter, Bareiss and Holte, 1990) is a case-based classification system applied to the clinical audiology domain. The system uses on exemplar-based learning (Kibler and Aha, 1987; Bareiss, Porter and Wier, 1988; Branting, 1989; Aha, 1995), a method especially useful for domains lacking a strong domain theory. In the system, classification is combined with explanation. Explanation is used for justifying case classification and for determining similarity between two cases.

Cunningham et al. have presented an incremental retrieval system, I-CBR (Cunningham, Bonzano and Smyth, 1995). For this case-based reasoning paradigm, the user specifies a case skeleton and attempts to retrieve all cases similar to it. I-CBR partitions attributes for a case into two groups

– the ones which are known immediately and the ones that need some extra effort for specifying them. During the retrieval process, the user specifies known attributes and the system returns a pool of case candidates. From the set of unknown attributes the most discriminating one is selected and the user is queried for its value. Thus, the initial pool of retrieved cases is processed to decrease its size to eliminate less-similar cases, which increases classification accuracy.

A.6.6 Case-Based Control

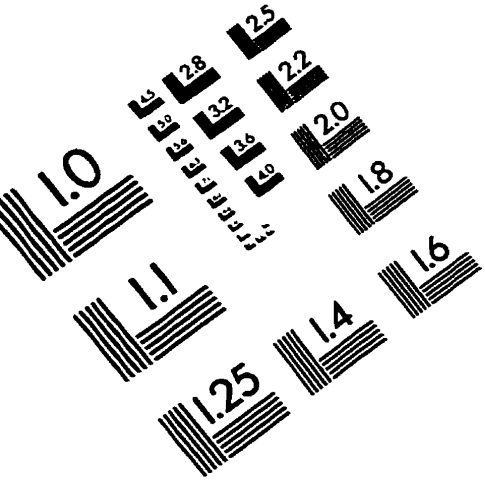
A control system capable of remembering a repeated operation is called a learning control system (Arimoto, Kawamura and Miyazaki, 1984; Bondi, Casalino and Gambardella, 1988; Wang, Soh and Cheah, 1995). A learning process can increase the number of solvable tasks (i.e., acquire new knowledge) or it can improve the efficiency of solving repeated tasks. Most current applications of machine learning in robotics are oriented to the perfection of the control process. Naniwa and Arimoto proposed a learning control system for manipulators whose endpoint is moving under geometrical constraints on a surface (Naniwa and Arimoto, 1995). (Jang, Choi and Ahn, 1995) describes an iterative learning system for precise tracking control with guaranteed convergence.

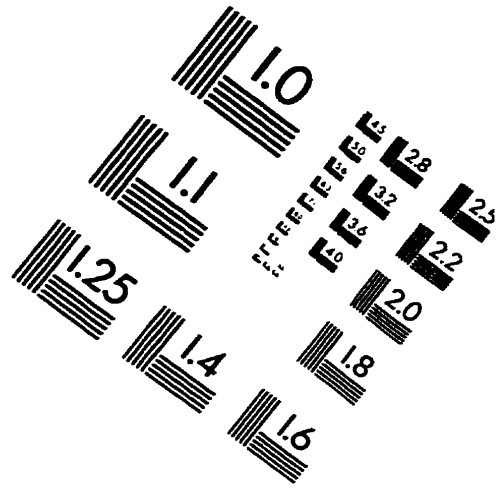
The Theo-Agent is a reactive system (Mitchell, 1990). The system is used in robot control and combines a stimulus-response subsystem for rapid reaction with a search-based planner for handling unanticipated situations and an explanation-based learning mechanism to construct stimulus-response rules to cover new situations. Thus, the system becomes increasingly reactive, since over time, less planning is required as the machine learning component acquires more control rules.

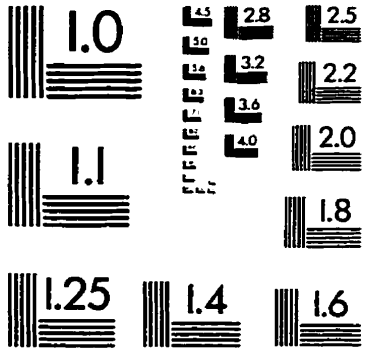
The Candide system (Luzeaux and Zavidovique, 1994) has been used for the acquisition of control knowledge. The system learns a qualitative model of the system first; then, on the basis of the mode, it creates a rule-based incremental controller that controls a given system.

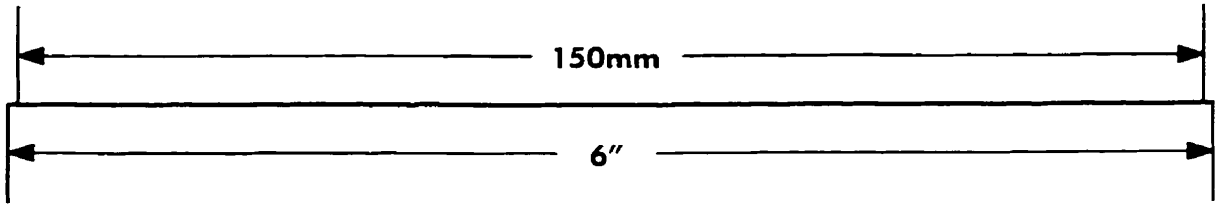
The SINS system (Ram and Santamaría, 1993a) is a self-improving reactive control system for autonomous robotic navigation. The system is based on using case-based techniques and reinforcement learning. The learning module monitors the system and incrementally modifies the case representations to accommodate the changes. Because of the utility problem (Minton, 1988b), the SINS system uses only 10 cases to allow for real-time performance.

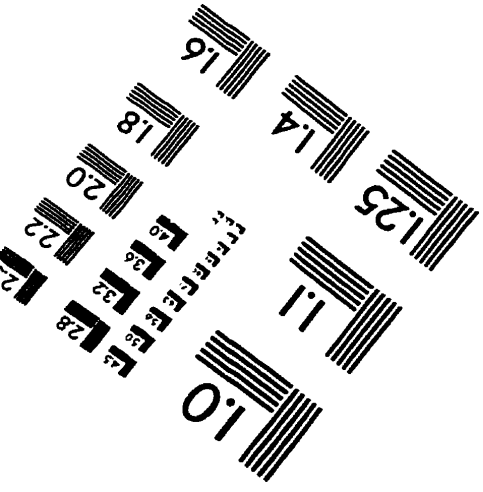
IMAGE EVALUATION
TEST TARGET (QA-3)

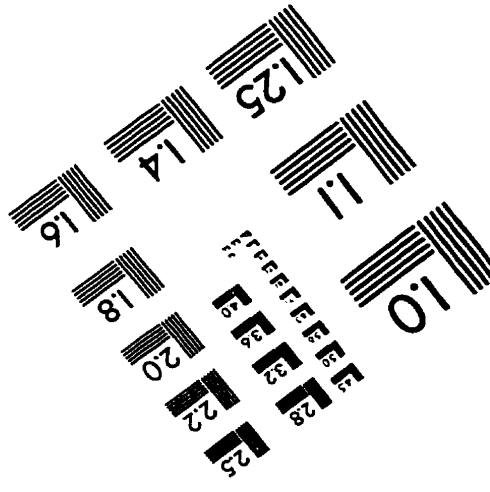












APPLIED IMAGE, Inc
1653 East Main Street
Rochester, NY 14609 USA
Phone: 716/482-0300
Fax: 716/288-5989