Case Retrieval Nets as a Model for Building Flexible Information Systems

Dissertation

zur Erlangung des akademischen Grades **Dr. rer. nat.**
im Fach **Informatik**

eingereicht an der

Mathematisch-Naturwissenschaftlichen Fakultät II
der Humboldt-Universität zu Berlin

von

Diplom-Informatiker Mario Lenz
geb. 26. Dezember 1967 in Berlin

Präsident der Humboldt-Universität zu Berlin
Prof. Dr. Dr. h.c. Hans Meyer

Dekan der Mathematisch-Naturwissenschaftlichen Fakultät II
Prof. Dr. sc. nat. Bodo Krause

Gutachter

1. Prof. Dr. Hans-Dieter Burkhard
2. Prof. Dr. Michael M. Richter
3. Prof. Kevin Ashley
4. Prof. Dr. Bernd Neumann

Tag der mündlichen Prüfung: 16. Juli 1999

Abstract

In this thesis, a specific memory structure is presented that has been developed for the retrieval task in Case-Based Reasoning systems, namely Case Retrieval Nets (CRNs). This model borrows from associative memories in that it suggests to interpret case retrieval as a process of *re-constructing* a stored case rather than *searching* for it in the traditional sense. Tow major advantages of this model are efficiency and flexibility:

- Efficiency, on the one hand, is concerned with the ability to handle large case bases and still deliver retrieval results reasonably fast. In this thesis, a formal investigation of efficiency is included but the main focus is set on a more pragmatic view in the sense that retrieval should, in the ideal case, be fast enough such that for the users of a related system no delay will be noticeable.

- Flexibility, on the other hand, is related to the general applicability of a case memory depending on the type of task to perform, the representation of cases etc. For this, the concept of information completion is discussed which allows to capture the interactive nature of problem solving methods in particular when they are applied within a decision support system environment. As discussed, information completion, thus, covers more specific problem solving types, such as classification and diagnosis.

The formal model of CRNs is presented in detail and its properties are investigated. After that, some possible extensions are described.

Besides these more theoretical aspects, a further focus is set on applications that have been developed on the basis of the CRN model. Roughly speaking, two areas of applications can be recognized: electronic commerce applications for which Case-Based Reasoning may provide intelligent sales support, and knowledge management based on textual documents where the reuse of problem solving knowledge plays a crucial role. For each of these areas, a single application is described in full detail and further case studies are listed for illustration purposes. Prior to the details of the applications, a more general framework is presented describing the general design and implementation of an information system that makes uses of the model of CRNs.

Zusammenfassung

Im Rahmen dieser Arbeit wird das Modell der Case Retrieval Netze vorgestellt, das ein Speichermodell für die Phase des Retrievals beim fallbasierten Schliessen darstellt. Dieses Modell lehnt sich an Assoziativspeicher an, insbesondere wird das Retrieval als *Rekonstruktion* des Falles betrachtet anstatt als eine *Suche* im traditionellen Sinne. Zwei der wesentlichen Vorteile des Modells sind Effizienz und Flexibilität:

- Effizienz beschreibt dabei die Fähigkeit, mit grossen Fallbasen umzugehen und dennoch schnell ein Resultat des Retrievals liefern zu können. Im Rahmen dieser Arbeit wird dieser Aspekt formal untersucht, das Hauptaugenmerk ist aber eher pragmatisch motiviert insofern als der Retrieval-Prozess so schnell sein sollte, dass der Benutzer möglichst keine Wartezeiten in Kauf nehmen muss.

- Flexibilität betrifft andererseits die allgemeine Anwendbarkeit des Modells in Bezug auf veränderte Aufgabenstellungen, auf alternative Formen der Fallrepräsentation usw. Hierfür wird das Konzept der Informationsvervollständigung diskutiert, welches insbesondere für die Beschreibung von interaktiven Entscheidungsunterstützungssystemen geeignet ist. Traditionelle Problemlöseverfahren, wie etwa Klassifikation oder Diagnose, können als Spezialfälle von Informationsvervollständigung aufgefasst werden.

Das formale Modell der Case Retrieval Netze wird im Detail erläutert und dessen Eigenschaften untersucht. Anschliessend werden einige möglich Erweiterungen beschrieben.

Neben diesen theoretischen Aspekten bilden Anwendungen, die mit Hilfe des Case Retrieval Netz Modells erstellt wurden, einen weiteren Schwerpunkt. Diese lassen sich in zwei grosse Richtungen einordnen: intelligente Verkaufsunterstützung für Zwecke des E-Commerce sowie Wissensmanagement auf Basis textueller Dokumente, wobei für letzteres der Aspekt der Wiederbenutzung von Problemlösewissen essentiell ist. Für jedes dieser Gebiete wird eine Anwendung im Detail beschrieben, weitere dienen der Illustration und werden nur kurz erläutert. Zuvor wird allgemein beschrieben, welche Aspekte bei Entwurf und Implementierung eines Informationssystems zu beachten sind, welches das Modell der Case Retrieval Netze nutzt.

Mario Lenz
Berlin, 16. Juli 1999

Summary

This thesis presents a specific memory structure that has been developed for the retrieval task in Case-Based Reasoning systems. This memory structure borrows from associative memories in that it suggests to interpret case retrieval as a process of *re-constructing* a stored case rather than *searching* for it in the traditional sense of AI. Thus, case retrieval should be viewed as a *bottom-up* instead of a *top-down* process.

After a brief introduction to the most fundamental concepts of Case-Based Reasoning, the idea of viewing problem solving as *information completion* processes is described. By doing so, the focus of this work is clearly set on the engineering point of view, that is on building systems that help to solve problems. Cognitive aspects, on the other hand, only play a minor role.

The concept of *information completion* appears to be particularly useful for decision support systems, i.e. in situations when a fully automatic system is not appropriate (or not a realistic goal). However, it cannot be implemented by means of more traditional retrieval techniques. The shortcomings of these techniques are discussed and criteria are established which should be satisfied by a retrieval method for decision support systems. Very briefly, these criteria are:

- Case retrieval should be highly efficient in order to build systems that can cope with large case bases for real-world applications.
- Case retrieval has to be flexible in terms of the types of queries that are supported and the similarity functions that can be implemented.
- The case memory on which the retrieval method is working should be easy to maintain. In particular, updates in order to reflect changes in the underlying case data should be easy. But, likewise, modifications to the similarity model should be easy to incorporate.

A major result of this thesis is the model of Case Retrieval Nets which satisfies the above criteria and has been successfully implemented in a number of applications. The model is described in full detail in a separate chapter, including its properties, advantages, and shortcomings. Also, possible extensions are presented and their relationships to the basic model are discussed.

Besides these more theoretical aspects, a further focus is set on applications that have been developed on the basis of the Case Retrieval Net model. Some of these applications are deployed systems that are today used in commercial settings. Of course, for fielding such applications much work is required beyond Case-Based Reasoning itself. Nevertheless, the success of these systems proves that the models developed as part of this thesis not only work in theory but can also cope with the rigorous and sometimes peculiar requirements of industry.

The range of applications covered in this thesis can be broadly classified into two categories: Firstly, in electronic commerce situations intelligent support is needed when customers search for products which suit their needs. Secondly, for customer support tasks techniques related to knowledge management are developed. For each of these two categories an application is described in full detail; for the latter even a new sub-area of CBR, *Textual CBR*, has been established recently by a number of research sites. Other projects which appear to be closely related to these are only sketched briefly.

In addition to the concrete applications, a general framework is presented showing how, in general, a decision support system based on CRNs should be composed, designed, and implemented. The detailed descriptions of the above mentioned applications can be seen as *case studies* of how this framework can be implemented.

Due to the strong focus on applications, other topics that might be of interest are not sufficiently addressed in this thesis. In particular, the cognitive aspects of Artificial Intelligence in general and Case-Based Reasoning only play a minor role despite Case Retrieval Nets showing some properties which suggest interesting relationships to cognitive models.

Acknowledgments

My thanks go out to a large number of people. First of all, I would like to thank Prof. Dr. Hans-Dieter Burkhard for many years of fruitful cooperation. In 1993, he offered me the chance to participate in the research pursued in the area of Case-Based Reasoning, to actively attend one of the first major events in this area, the First European Workshop (EWCBR-93) which was held in Kaiserslautern in 1993, and to write my master's thesis on this subject. After that, I gladly accepted the offer of joining his research group as a member of the research staff in early 1995. From that time on, he has been an invaluable source of ideas. Working in his group I especially appreciated his style of discussing ideas, giving hints, and then letting people wander off their own way. This, obviously, very much supports people learn to think for themselves.

In particular, the work presented in this thesis emerged from a discussion that we had immediately after EWCBR-93 in which Prof. Burkhard described problems he saw with traditional case retrieval methods and how concepts from cognitive psychology might be used for building an alternative model. Based on these initial ideas, the model of Case Retrieval Nets developed over time. In so far, he has been involved in most parts of the work presented here although his name is not always explicitly present.

My special thanks also go to Prof. Michael M. Richter from the University of Kaiserslautern as well as to Prof. Kevin D. Ashley from the University of Pittsburgh for the discussions we had at several occasions. I am thankful for all the useful suggestions and critical remarks.

Major parts of the work presented here would not have been possible without the help of a number of colleagues and students who have been working in the various projects over the years: André Hübner, Mirjam Minor (formerly Kunze), Marko Schrenker, Gordon Kramer, Marlies Gollnick, and (more recently) Peter Hammels. All these people have been engaged in the development of the systems described in this thesis and, thus, contributed to the successful applications of the technology.

Furthermore, most applications have been performed in cooperation with industrial partners. As described, we joined forces with *TecInno*, Kaiserslautern, for most of these projects. Here, I am indebted to Thomas Roth-Berghofer and Joachim Schwalb who both helped making useful tools from the first prototypes. And, of course, thanks goes to Stefan Wess who took the risk of developing industrial applications in cooperation with our university group and who, during his years at the University of Kaiserslautern, helped with a lot of helpful comments and suggestions. In fact, he being one of the main organizers of EWCBR-93 very much influenced and motivated me during the first time within the community.

Also, the projects very much benefited from the help of the actual users who showed plenty of patience and were willing to answer the many questions that occurred during the various phases of system development. In particular, I want to single out by name: Karl-Heinz Busch from the SIMATIC Online Support at *Siemens* as well as Michael Posthoff and Roman Borch from *check out*.

I also want to thank my colleagues who have been working in different areas but nevertheless supported me during recent years: Renate Zirkelbach for dealing with the organization of so many things and thus keeping me free of all this paper work; Olga Schiemangk for always being willing to help when problems with the machines occurred; Gabriela Lindemann and Ralf

Kühnel for relieving me from some of the work concerned with lectures during the final phase of writing this thesis.

Last but not least thanks goes to a number of people with whom I have discussed many aspects of this work as well as Case-Based Reasoning, in general, in recent years, such as Ralph Bergmann, Wolfgang Wilke, Brigitte Bartsch-Spörl, Katy Börner, Max Wolf, Eric Auriol, and Michel Manago.

I know there were others who have helped me along the way and I apologize for not naming them here.

Mario Lenz
Berlin, July 1999

Contents

List of Figures

List of Tables

Part I

Background

Chapter 1

Introduction

Human experts are not systems of rules, they are libraries of experiences.
Riesbeck and Schank, Inside Case-Based Reasoning, 1989

This introductory chapter gives a motivation for the research pursued and describes the background in which the work was settled. It states the objectives, summarizes the main results, and gives an outline of the entire thesis.

In recent years, the Case-Based Reasoning (CBR) paradigm has gained much attention both in research and industry. Broadly speaking, CBR is concerned with solving new problems by remembering and adapting solutions that have worked in the past. Thus, CBR attempts to simulate the use of experiences by human problem solvers:

> When confronted with a specific problem, we often relate the problem to situations we have experienced previously. We *remember* situations in the past that have been *similar* to the problem at hand and we try to *adapt* and *reuse* the knowledge from the earlier situation. The result of this process will then become part of our experience. Thus, we can make direct use of lessons learned in the past rather than reasoning from first principles every time.

After the fundamental ideas had been developed and first workshops had taken place in the USA in the late 1980's, a strong and relatively independent community developed in Europe. From these activities, a large number of research projects evolved which lead to new theoretical insights. Today, only a decade after the first workshops, CBR has its place in the field of AI, with a number of relationships to other areas (such as Machine Learning and statistics), contributions to all major AI events, its own conferences and workshops, as well as books addressing different aspects of the CBR technology.

Apart from the theory, however, CBR research in Europe has always been very much application-oriented. As a consequence, a multitude of applications has been developed ranging from academic prototypes to deployed industrial systems. In Section 2.1.3 we will briefly mention some of these. It is obvious that both the rapid deployment of these applications as well as the long term success of the CBR paradigm require, among other things, a sound theoretical grounding. In this dissertation, one specific theory, the model of *Case Retrieval Nets* will be presented. As we will see, this theory provides both new insights into case retrieval and the notion of similarity as well as a model for efficient case retrieval which has been successfully implemented in a variety of applications.

1.1 Thesis Overview

As with Artificial Intelligence, in general, Case-Based Reasoning, too, is concerned with two different but related facets of the notion of intelligence:

- A first motivation is to establish cognitive models in order to understand *human* thinking and behavior. How much CBR is tied to this type of research, becomes obvious when considering the history of CBR which has, in fact, its roots in cognitive psychology.
- A second motivation is to build systems which help to solve real-world problems. Here, the so-called *cognitive adequacy* is of minor importance. Rather, systems have to be implemented which show some typical performance elements of intelligence (such as flexibility, adaptability to changing environments etc.) while others are explicitly ignored (for example, creativity or even forgetting).

When building intelligent systems, formal descriptions are required. For this, mathematical models seem most appropriate. Compared to other AI methods, however, Case-Based Reasoning appears to be a certain kind of *methodology* rather than a specific *technique*. For example, rule-based systems have a well-defined set of components (namely facts, the rule base, and a rule interpreter) as well as a precisely defined model of how these different components interact (see, for example, Nilsson 1982, Chapter 6 as well as Gonzalez and Dankel 1993). Consequently, the behavior of a rule-based expert system shell can only be influenced by specifying an appropriate set of rules. Once the rules are defined, the behavior of the entire system is fixed. So, the only way of bringing in domain-specific knowledge is via the rule set.

The situation is a bit different for CBR systems: In addition to the cases that have to be provided, a similarity model, means of adaptation, and retainment functionality etc. have to be specified, too. As all these components strongly depend on the targeted application, every implemented CBR system has its specific properties which clearly distinguish it from other systems.

Consequently, CBR is positioned along the borderline between cognitive science, mathematical theories, and practical systems design as a software engineering discipline. This often causes difficulties with respect to how much one or the other aspect of these areas may be neglected. Typical questions are:

- To what extend can cognitive models be ignored for the implementation of practical systems? Is it required that an artificial system models the way human beings think or can there be different ways leading to the same goal[1]?
- In how far are mathematical models adequate for describing the requirements in a particular application area as well as the behavior of a specific system?

This thesis clearly addresses issues related to the *engineering point of view*. That is, the objective consists in the development of special techniques that appear to be highly useful for a number of tasks a CBR system might have to deal with. During a number of discussions, we also realized that there might be interesting contributions with respect to cognitive models — however, this topic has not (yet) been explored further.

The problems related to an exact mathematical model underlying a CBR system in general will be illustrated in Chapter 3 when we discuss a notion of similarity which is somehow different to distances and metrics as usually applied in mathematics.

1.1.1 Background

The work reported about in this thesis has its origins in a number of projects performed in the research group of Prof. Hans-Dieter Burkhard at the Department of Computer Science

[1] Although wheels are really an invention of mankind, they perform better than legs in specific environments.

of Humboldt University, Berlin. This research dates back to the early 1990's when the first CBR activities have been launched in Germany. The author himself has been active in the area since 1993 when he implemented the CaBaTA[2] system (Lenz 1994b) to illustrate the general principles of Case-Based Reasoning in an easy to understand domain, namely decision support when searching an appropriate holiday tour package (Lenz 1993; Lenz 1994a; Lenz 1996b).

Since that time, research has spread out to diverse directions and a number of other projects have been performed. That's why, parts of this thesis have already been published when reporting about specific results:

- The starting point for this thesis was the CaBaTA system itself.
- Building on CaBaTA, some other application areas have been explored, as for example real estate assessment and brokerage of apartments in Berlin (Lenz and Ladewig 1996).
- Already during these early research projects a new kind of memory model for case retrieval has been developed which did not rely so much on a traditional search through the case base but formed a starting point for the model of Case Retrieval Nets as they will be presented in this thesis (Lenz and Burkhard 1995).
- Within FAllQ, another application project that started in late 1996, the requirements were changed from handling of structured data to the management of textual information (Lenz and Burkhard 1997b). While still applying the methodology of Case-Based Reasoning, a number of specific problems had to be solved here. After having found solutions for most of the related problems, the CBR-Answers system has been implemented for the specific requirements of Textual CBR. In March 1998, Siemens installed the SIMATIC Knowledge Manager for providing customer support for their SIMATIC products over the internet[3].
- The Virtual Travel Agency[4] project was launched in Spring 1997 with the objective of combining CBR and Internet technology in an electronic commerce environment in order to provide all kinds of services offered by a travel agent via the World Wide Web.

  In the very heart of the Virtual Travel Agency is a CBR system searching an electronic product catalogue for all available Last Minute tour packages from a major German tour provider. Today, the Virtual Travel Agency is among the most successful Internet applications for the travel industry in Germany. At the time of writing this thesis, that project still continues with the installation of additional services.

The work reported about here, in particular all descriptions related to the Case Retrieval Net model and its applications, arose from these projects. In that sense, this thesis summarizes the models developed, the experiences collected, and the procedures implemented for all these different application areas.

As becomes obvious from the above listed projects, our research has always been very much application- and demand-driven. That is, in most cases we started with a specific problem at hand and extended our technology in such a way that we were able to handle these tasks. Furthermore, we did not stop once a *principle* solution has been developed. Rather, due to projects in industrial settings we were forced in most cases to continue with our work until a *working* implementation had been delivered.

We think this is slightly different from many other research sites. Also, this might explain some of the problems we had to deal with (such as extremely short development cycles) as well as the priorities that were set up. Concerning the latter point we chose to consider the

[2] Case Based Travel Agency
[3] http://www4.siemens.ad.de/skm
[4] http://www.reiseboerse.com

user requirements more important than, say, pure academic goals. This resulted in approaches to some of the problems which differed from the more traditional way they are addressed by research groups — as for example when evaluating the results of experimental studies (cf. Section 9.4).

1.1.2 Focus

As discussed further in Section 2.2, a CBR system in general has to deal with a wide variety of tasks ranging from case representation to retrieval, adaptation, and learning. Obviously, covering all these aspects in depth is much too complex to be handled in a single thesis.

Aspects of Decision Support

Instead, we will focus on issues of case representation and case retrieval. This is motivated by the above sketched application projects which can all be subsumed under the heading of *decision support* systems: Due to the complex nature of these applications, it seemed neither promising nor desirable to build autonomous problem solvers. From what we have learned from the performed projects, we conclude that customers can often be satisfied with a system that is designed to *support* the human users by providing, maintaining, and distributing relevant information instead of *replacing* them (cf. also Section 2.2.6).

We argue that in highly complex domains providing relevant information may already be of great help to human experts when trying to solve problems. Such a system then plays the role of an *external memory* (Burkhard 1995a) or, as it is called nowadays in the context of knowledge management, an *organizational memory* (Abecker, Bernardi, Hinkelmann, Kühn, and Sintek 1998; Abecker, Decker, and Kühn 1998). This means, that the primary tasks of such a decision support system are

- to *collect* the knowledge of all the experts working in the field;
- to *preserve* this knowledge in a way that allows it to be used for future tasks;
- to *disseminate* the knowledge by allowing for an efficient and flexible access in a particular problem context.

As we will show, Case-Based Reasoning is a promising technique for implementing these types of systems. This is true despite issues of adaptation, integrated learning etc. not playing a major role in the design of such a system.

Our interpretation is that in the context of decision support, both the human expert *and* the IT system together form what is usually considered to be the problem solver. Then, adaptation, for example, is performed by the human expert based on the information provided by the system — hence *within* the problem solver.

> **Example 1.1** Consider an IT system that stores documents about a specific range of products and makes them accessible via some intelligent search engine: When highly skilled technicians query the system when trying to repair some devices, these experts are well able to interpret the information provided in a small number of documents and to check whether the hints given fit their particular problem situation. However, they are usually not able to have in mind all the information about the entire product line, including all the latest updates. Providing this information in an efficient and flexible manner is exactly the task of the IT component.

1.1.3 Objectives

As mentioned above, this thesis is concerned with the implementation of intelligent systems based on the Case-Based Reasoning paradigm. Moreover, we will focus very much on the aspects of case representation and retrieval. Since we want to address real world problems, CBR has to provide solutions for a number of problems in order to allow for building successful applications. We will in particular address problems of *efficiency*, *flexibility*, and a development *methodology*.

Efficiency

In order to become a successful technology for real-world applications, Case-Based Reasoning has to be able to handle large case bases efficiently. We will take here a pragmatic view on efficiency: Even with many cases, the system still has to respond in reasonable time. What, precisely, that means is again highly application-dependent but it should be clear that case bases of several hundreds of cases as they have been reported a few years ago in the scientific literature, e.g. by Goos (1994), are not realistic. Rather, recent developments, for example in the area of electronic commerce require the handling of several thousands of products.

> **Example 1.2** In the VIRTUAL TRAVEL AGENCY application described in more detail in Chapter 8, the case base consists of up to 250,000 offers during peak season. As the entire system is WWW-based, an immediate response by the system is required, that is retrieval time should be no more than a second.

It is a crucial requirement that CBR systems working in those areas still provide an efficient retrieval mechanism in order to make the overall approach feasible.

Flexibility

Another advantage that CBR technology has to exploit is that such systems may be designed much more flexible than systems based on more traditional technologies. In particular, we consider here flexibility in terms of

Case representation: In many areas, encoding cases by means of predefined case formats, such as feature vectors, is not an appropriate approach. Rather, each case may have specific components and, hence, size and structure differ from other cases.

Case retrieval: Not only should case retrieval be highly efficient but also flexible enough to cope with the user's needs. In the VIRTUAL TRAVEL AGENCY domain, for example, some customers may search for a suitable destination for a given budget and certain preferences while other customers would like to enter the destination and query the system about available climate and pricing information.

System design: In order for CBR to become a successful technology on the market, it is not sufficient to implement some isolated systems. Rather, techniques are required which are easily adaptable for varying environments — be it size of the case base, complexity of single cases, properties of the similarity measure and so on.

Methodology

In order for CBR to become more widely used, it is crucial that domain experts who, however, only have a rough idea of what CBR is and how it works are enabled to build their own CBR-based applications. Of course, they will hardly ever start from scratch and really implement a completely new system. Rather, they will most likely consider the tools available on the market, check whether these are appropriate for the particular problem domain, and configure the chosen tool to fit the requirements.

For this, a kind of *framework* is required which clearly describes

- for what types of applications a particular tool has been designed;
- what are the requirements for a (successful) deployment;
- what has to be done to obtain an optimal performance of the system, i.e. what are the *knowledge containers* that the user of such a tool should think about in order to obtain the best possible results.

We will address these issues in Chapter 7 from a general perspective while the subsequent application-oriented chapters present case studies of how this framework was applied in various applications.

As we will discuss in Chapter 3, more traditional CBR techniques have severe problems with at least one of the above requirements. Consequently, there is an urgent need for a model which can cope with all three aspects.

1.1.4 Results

The major results of this thesis can be summarized with respect to two different aspects: Firstly, a model for case memory has been developed which appears to be highly suitable for the task of case retrieval in decision support systems. In particular, it fulfills the goals set in the previous section in that it is highly efficient, flexible, and easy to maintain. Secondly, it has been explored how this formal model can be utilized for building decision support systems and how the specific requirements of different applications can be dealt with.

We think that the following results are particularly important:

1. Problem solving should be seen as an information completion process. This is in particular true for domains where decision support is the only realistic goal rather than a fully automatic system. It is shown that more traditional problem solving methods, such as classification, cannot support such interactive tasks (Section 3.1).

2. Case-Based Reasoning is an appropriate technology for implementing the process of information completion even though a sophisticated inference is not necessarily performed within the system itself. In a certain sense, the computerized retrieval system *and* the expert together form the reasoning system (Section 3.3).

3. The model of Case Retrieval Nets fulfills the requirements defined in the previous section in that it is highly efficient, flexible, and easy to maintain. These properties can be shown formally (Section 5.3). Also, this model is open for a variety of extensions which might ease the utilization of that model for specific applications (Chapter 6).

4. A number of applications have been developed in which Case Retrieval Nets are an essential component. These applications range from domains with structured case representations, such as in electronic commerce situations (Chapter 8), to the handling of textual documents, such as in hotline scenarios (Chapter 9). Case Retrieval Nets appear to be flexible enough to cope with the different requirements that were defined by each of these applications. Also, in some of the deployed applications, case bases much larger than in many other CBR systems are in use without any performance problems.

1.1.5 Applications

The technology that is the topic of this thesis has been successfully applied in a number of applications with varying characteristics. Generally speaking, all these applications are decision

support systems (see Section 2.2 for a more detailed explanation). When investigating the applications in detail, however, each of them shows characteristic properties.

Nevertheless, the applications presented in Chapters 8 and 9 can be grouped in two main areas:

- On the one hand, there are applications dealing with the retrieval of information from a case memory where this information encodes (in a more or less structured form) properties of products and the like. On a very general level, these applications could be subsumed under the heading of *electronic product catalogues* and information systems supporting electronic commerce.

- On the other hand, there are systems handling textual documents. These are *Textual CBR* systems which aim at managing the information contained in semi-structured documents and providing means for content-oriented retrieval.

These two application areas should be considered as two extremes rather than disjoint categories. Of course, textual descriptions may also provide valuable information in electronic product catalogues. Likewise, the Textual CBR applications make use of some structured information. There is, however, a difference concerning to what extend structured information can be used. From a pragmatic point of view, a further difference between the two is that the former is primarily used in *pre-sales* and *sales* situations whereas the latter deals with *after-sales* support in electronic commerce situations (Wilke, Lenz, and Wess 1998).

E-Commerce–Oriented Applications

VIRTUAL TRAVEL AGENCY**:** An application making a broad spectrum of all the services provided by a travel agent accessible via the World Wide Web. The main component, where CBR plays a crucial role, is a system providing intelligent access to Last Minute tour packages of one of the largest German tour providers. The system has been in use since Spring 1997 and is a commercial success. It is described in detail in Chapter 8.

REAL ESTATE ASSESSMENT**:** A system applying Case-Based Reasoning to the task of real estate assessment and brokerage of apartments. This project has been performed in cooperation with a German bank but has only been followed up to a prototypical implementation. See Section 8.3.1 for more details.

CBR-SELLS**:** An approach of providing generic CBR capabilities in the context of electronic commerce applications is the CBR-SELLS system developed in cooperation with *TecInno*. The idea here is to provide a library that can be used as a *plug in* by developers of electronic shops. Thus, the capabilities of such shops can be extended by case-based technologies while still providing users the traditional interfaces — however with the database-oriented retrieval being replaced by a case-based retrieval. CBR-SELLS very much relies on specific aspects of this type of applications, such as a representation of products based on attribute–value pairs and a more or less straightforward similarity model. CBR-SELLS will be described in Section 8.3.2.

Textual CBR Applications

SIMATIC KNOWLEDGE MANAGER**:** A commercially used system supporting hotline staff at Siemens *Automation & Drives*. Case-Based Reasoning is utilized for tasks of knowledge management in the sense of making diagnostic knowledge available to technicians dealing with specific *Siemens* products. The specific properties of this application are explained in Chapter 9.

FALLQ: Similarly to the SIMATIC KNOWLEDGE MANAGER, the FALLQ project addressed issues of knowledge management, however the main focus here was to provide tools for managing the *in-house* knowledge, i.e. to provide access to textual information across distributed project teams. See Section 9.5.1 for details.

EXPERIENCEBOOK: This application, too, addresses problems of knowledge contained in textual documents but this time with a more technical focus, namely the support of system administrators at the Department of Computer Science at Humboldt University, Berlin. The EXPERIENCEBOOK is explained in Section 9.5.2.

Purpose of Applications

The main objective of research presented in this thesis was the development of CBR techniques that could handle the problems listed in Section 1.1.3. Consequently, the applications are primarily used for demonstrating that these objectives have been achieved. On the other hand, a full implementation of some of these applications would require much more work in completely different areas. This has *not* been the topic of this thesis and in this sense some of the presented applications may be considered as *prototypical* while others are fully deployed in industrial settings.

1.2 Prerequisites

This thesis is not intended to be an introductory textbook to CBR. Rather, the objective is to present a specific model of how a CBR system might be implemented, to discuss the pros and cons of this model, and to summarize the lessons learned from projects.

In order to clarify the vocabulary being used and to prepare the ground for the subsequent chapters, we will give a very brief introduction to CBR in Chapter 2. However, to obtain a more profound understanding of CBR in general, the reader is referred to some of the books which (at least in some chapters) address CBR on a more introductory level (Kolodner 1993; Wess 1995; Lenz, Burkhard, Bartsch-Spörl, and Wess 1998).

1.3 Guide to the Thesis

This thesis consists of four parts each with a different focus. The first part gives a brief (and very shallow) introduction to CBR and presents a specific view on problem solving processes. The second part describes in full detail the model of Case Retrieval Nets and possible extensions. The third part then addresses applications and the fourth discusses the work presented.

- To objective of **Chapter 2** is to present the most fundamental concepts of Case-Based Reasoning and to introduce the vocabulary that will be used later on.
- The idea of viewing problem solving in decision support systems as information completion processes is presented in **Chapter 3**.
- **Chapter 4** describes the starting points in the development of the Case Retrieval Net model. In particular, the traditional way of viewing retrieval as a search through case memory is discussed.
- The formal model of Case Retrieval Nets is presented in **Chapter 5**, including properties of that model.
- In **Chapter 6**, a number of possible extensions to the basic model are discussed some of which have been fully implemented while others have only been used in prototypical systems.

- After the theoretical concepts have been introduced, **Chapter 7** addresses the question of how a specific application can be build using Case Retrieval Net technology. It is shown which components are required and how these should interact.

- **Chapter 8** gives a detailed description of a very successful application, the VIRTUAL TRAVEL AGENCY, that has been built by means of CRNs. The chapter also discusses, more briefly, some other projects situated in an electronic commerce setting.

- Another type of application is addressed in **Chapter 9**. Here, the objective is to reuse knowledge contained in specific technical documents, such as collections of *Frequently Asked Questions*. The chapter presents some techniques which are particularly suited for Textual CBR. Another successful system, the SIMATIC KNOWLEDGE MANAGER is described in full detail, while related projects are only sketched.

- Related work is discussed in **Chapter 10**. In particular, work from such diverse areas as Case-Based Reasoning, Information Retrieval, Knowledge Management, and cognitive psychology are addressed and their relationships to the work presented in this thesis are shown.

- **Chapter 11** concludes with a summary of the thesis and gives an outlook to the work in the near future.

Chapter 2

A Brief Introduction to Case-Based Reasoning

In order to solve a new problem, one should first try using methods similar to those that have worked on similar problems.
Marvin Minsky, Computers and Thought, 1963

In this chapter, we will give a brief introduction into the basic concepts of Case-Based Reasoning. This is not intended to be a comprehensive introductory text to all the aspects related to the topic. Rather, the objective is to recall some of the most fundamental concepts of CBR which will be required for an understanding of the following chapters. The reader familiar with CBR may thus skip the chapter.

2.1 Origins of CBR

2.1.1 Normative Approaches in AI

During the first decades of its existence, Artificial Intelligence has been directed very much towards rule-based and model-based approaches. Researchers attempted to construct *normative* theories for both the description of human problem solving behavior as well as the implementation of intelligent systems. This is true for both research in cognitive science (Anderson 1983) as well as the attempts to build intelligent systems (Hayes-Roth, Waterman, and Lenat 1983; Shortliffe 1976).

A principle assumption underlying these efforts was that intelligent behavior can always be described by means of *generalized* knowledge units which are independent of each other, such as rules in a rule-base. This assumption, however, causes a number of deficiencies with respect to both a theory of human thinking and the design of computer systems. Among these are:

- Generalized formalisms have problems when dealing with exceptions; that is, with situations that violate the general rule which, nevertheless, appears to be useful for most situations.
- When building a rule-based system, knowledge acquisition and knowledge execution are strictly separated. As a consequence, any such system will hardly be adaptable to changing environments.

- From a cognitive perspective, the separation of learning and problem solving is questionable, too. Rather, when solving problems, humans reuse experiences and, at the same time, acquire new ones.

In fact, these principle problems caused what is known today as the *crisis of expert system technology*. When attempting to build an expert system which solely relies on general knowledge, the following problems arise which, at the end, caused the failure of expert system approaches:

Knowledge-acquisition bottleneck: It is hard, if not impossible, to acquire the knowledge for a particular domain and to formalize it by means of independent but generally valid knowledge units.

Missing adaptability: Every problem is considered in isolation only, experiences from earlier problem solving episodes are not taken into account. Hence, no learning occurs.

Maintenance problem: As the acquired knowledge units are, in practice, not really independent of each other, a lot of implicit dependencies will be contained in the overall system — which prevents an efficient maintenance.

Growth of search space: As the knowledge units are assumed to be independent of each other, the search spaces which have to be explored during problem solving explode. Consequently, only relatively simple problems could be dealt with.

Brittleness: While systems based purely on a normative theory may be well suited for problems within a well-defined area, their problem solving competency drastically decreases when problems have to be solved that belong to the fringe of that area.

Taken together, all these problems make it difficult, if not impossible, to come up with a general framework and methodology for building intelligent systems which are applicable for a wide range of tasks. Nevertheless, normative models still play a major role in AI systems. The key point is that they should not be the only means of knowledge representation. Cases representing snapshots of concrete problem solving episodes are an alternative.

2.1.2 Historic Roots of CBR

Historically, three different areas of research contributed to the development of CBR:

- theories about story understanding and dynamic memory,
- analogical reasoning, and
- the development of decision support systems in jurisprudence.

The term *Case-Based Reasoning* itself has been coined by Roger C. Schank and Janet L. Kolodner which developed a theory of how human beings understand stories by putting them into context and explaining them by using the knowledge already existing (*Conceptual Dependency Theory*, Schank 1982).

Dynamic Memory

From the *Conceptual Dependency Theory*, the theory of *Dynamic Memory* evolved (Schank 1982; Kolodner 1983a; Kolodner 1983b) which claims that the three processes understanding, remembering, and learning all utilize the same memory structures. According to the *Dynamic Memory* theory, these memory structures can be represented as *Memory Organization Packets* (MOPs, Riesbeck and Schank 1989) which are structures containing information about how different components of the memory are related to each other.

This research at Yale University, eventually, lead to the first Case-Based Reasoning systems, such as CYRUS[1] (Kolodner 1983b).

Analogical Reasoning

Analogical Reasoning has a long tradition in both Artificial Intelligence and Cognitive Science. As with Case-Based Reasoning, the objective of Analogical Reasoning is to reuse knowledge from previous problem solving episodes directly, i.e. without generalization, for a new problem.

In Analogical Reasoning, a domain change usually occurs; that is, the current problem at hand is compared to problems from completely different domains.

> **Example 2.1** A good illustration for such a reasoning process is the analogy between the solar system (with the sun in the center and planets in the orbit) and an atom (with the nucleus in the center and electrons circling around it).

Obviously, there are common structures in both domains. These structures can be used to determine the similarity of the two problems, e.g. by using the *Structure Mapping Theory* developed by Gentner (1983).

In contrast, CBR systems are most often restricted to a particular domain and they attempt to reuse problem solving experiences from the same domain. Consequently, in both areas different topics were addressed in research: While CBR mainly dealt with case representation and case retrieval, Analogical Reasoning focussed very much on the transfer of knowledge in general and solutions in particular. A more detailed comparison of the two approaches has been given by Burstein (1989).

In recent years, the clear borderline between the two areas vanished. For example, the transfer of solutions is now also a major topic in CBR under the heading of *adaptation* (Kass 1991; Leake, Kinley, and Wilson 1995; Purvis and Pu 1995; Schmidt, Boscher, Heindl, Schmid, Pollwein, and Gierl 1995; Schumacher, Wilke, and Smyth 1998; Smyth and Keane 1996; Voß and Coulon 1996). Also, as we will see in later chapters, common aspects of case retrieval can be observed in that prior cases are *constructed* rather than *searched for* (Thagard and Holyoak 1989).

Decision Support for Jurisprudence

A third root of CBR, namely knowledge-based techniques handling jurisdiction problems, has been influenced and motivated not so much by cognitive issues but by practical needs. In particular in Anglo-American countries, the entire jurisprudence is based on precedents and the *Common Law* rather than well-defined laws as, for example, in Germany. Hence, the ability of finding an appropriate argumentation based on precedents is crucial in that domain.

Consequently, if a (knowledge-based) system is to be used to support the decisions of lawyers, it definitely has to take into account this specific nature of law and has to apply what is known today as Case-Based Reasoning (Ashley 1987; Ashley 1990; Rissland 1983; Rissland and Ashley 1987).

2.1.3 The History of CBR in Europe

With the beginning of the 1990's, European researchers became interested in Case-Based Reasoning, too. However, a slightly different direction was chosen here: While CBR in the United States has been very much influenced by cognitive science and, hence, cognitive aspects (such as cognitive adequacy and creativity) played a major role, this was not so much the case in Europe.

[1] Computerized Yale Retrieval System

Rather, right from the beginning European CBR research has been closely related to specific applications. As a consequence, a number of problems have been addressed which turned out to be severe obstacles for implementing systems in practice.

As an example for the different attitudes consider the problem of *efficiently* finding (i.e. *retrieving*) the most appropriate cases from memory:

- In the US community, the problem of case retrieval has been answered by assuming that it can be performed in parallel and, hence, if sufficiently many processors are available there is no efficiency problem (see, for example, Waltz 1989a; Domeshek 1989; Kolodner 1988b).
- In Europe, on the other hand, a number of techniques have been developed, each with specific advantages and drawbacks, which can be used for the task of case retrieval, for example kd-trees (Wess, Althoff, and Derwand 1993), the FISH & SHRINK model (Schaaf 1996), and the Case Retrieval Nets which are the topic of this thesis.

Due to this more application-oriented view on CBR, a number of projects evolved, including

- the FABEL project addressing industrial design (Gebhardt, Voß, Gräther, and Schmidt-Belz 1997; Börner 1998),
- the INRECA[2] project investigating the integration of CBR with inductive techniques (Althoff, Auriol, Bergmann, Breen, Dittrich, Johnston, Manago, Traphöner, and Wess 1995; Wess 1995),
- the INRECA-II project aiming at a general methodology for building CBR-based systems (Bergmann, Wilke, Althoff, Johnston, and Breen 1997; Bergmann, Breen, Göker, Manago, Schumacher, Stahl, Tartarin, Wess, and Wilke 1998),
- the recently launched WEBSELL project addressing intelligent sales support with Case-Based Reasoning, for example in an electronic commerce scenario (Wilke 1997; Wilke, Bergmann, and Wess 1998; Wilke, Lenz, and Wess 1998).

Apart from these (application-oriented) research projects, also a number of fielded applications have been developed; to name a few:

- *Analog Device*'s CBR server[3] for selling operational amplifiers (Wilke, Lenz, and Wess 1998; King 1998),
- the CASSIOPÉE system by *AcknoSoft* for troubleshooting and maintaining airplane engines (Heider 1996),
- the HOMER[4] system for supporting CAD/CAM users at *Daimler-Benz*' car development unit (Göker, Roth-Berghofer, Bergmann, Pantleon, Traphöner, Wess, and Wilke 1998),
- medical applications, such as GS.52 for the diagnosis of dysmorphic syndromes (Gierl and Stengel-Rutkowski 1994) and ICONS for antibiotic therapy advice (Gierl, Bull, and Schmidt 1998).

In the third part of this thesis, some more applications will be presented which range from prototypical implementations for demonstrating the feasibility of the CBR approach for the particular problem to deployed and highly successful applications — as in the case of the VIRTUAL TRAVEL AGENCY (Chapter 8) and the SIMATIC KNOWLEDGE MANAGER (Chapter 9).

[2] Induction and Reasoning from Cases
[3] http://www.analog.com
[4] Hotline mit Erfahrung - Hotline with experience

2.2 Basic Concepts of CBR

2.2.1 The General Idea of CBR

In a nutshell, the idea of using CBR for problem solving can be sketched as follows:

> In order to solve a particular *problem*, *search* for a *similar* problem that you have experienced in the past and try to *reuse* the *solution* of the past problem to the current situation.

This kind of problem solving behavior can be observed in every day life:

- A lawyer recalls precedents to argue in favor of the defendant.
- A physician remembers the case history of another patient.
- A mathematician tries to reuse a proof for a new theorem.
- A mechanic recalls that a particular strange noise once co-occurred with a broken crank shaft.
- A salesman remembers a previous customer who wanted a special product but could be satisfied with a slightly different one.
- A financial consultant has to decide about whether a customer is trustworthy or not and he recalls similar cases in which the loan had not been paid back.
- ...

Of course, the above idea is much too vague to allow for the building of a concrete model or the implementation of a system. In particular, there are key terms (which have been emphasized) that have to be clarified before we can proceed:

- What is a *problem*, what is its *solution*, and how are they represented and stored?
- How is the *search* for cases organized?
- When are two cases *similar*?
- How can an old solution be *reused*?

We will address these questions in the following. Some of the terms that will be introduced allow for a formal definition while others are expressions from everyday language. For the latter, we will just provide less formal descriptions and explanations paraphrasing the meaning of the term under consideration in sufficient detail. Should a formal definition of a term be required in later chapters, it will be given when needed.

2.2.2 Cases as Problems and Solutions

The idea of a case is to store a particular experience, an episode that might be worth remembering. In its simplest form, a case can be represented as a tuple consisting of a *problem* and a *solution*:

$$case = (problem, solution)$$

Thus, a case contains all the relevant information characterizing the problem situation and the solution, respectively. It does not, however, contain any generalized knowledge or details about lessons learned right now.

A query usually is considered as a specific case in that it represents the current problem situation, i.e. it does not yet have a solution component.

If useful, this fairly simplistic case format may be extended to include further components, such as the *outcome* of the case, i.e. whether applying the derived solution was a success or failure. As we will see in Chapter 3, it may even be helpful to not distinguish between a *problem* and a *solution* if it is not known a priori what part of a case will be reused as a problem description.

The case base is merely a set of cases. A case memory, in contrast, is a structured case base, i.e. a case base which, for retrieval purposes, is organized according to a particular memory structure.

2.2.3 Similarity of Cases

Similarity in Case-Based Reasoning is an attempt to model *utility*: How *useful* is a previous case for solving a current problem.

In fact, one has to struggle with a kind of paradox: A case should be considered as similar to the current problem, if the solution of that case can (easily) be reused for the problem. But, in general, this can only be determined *after* one has tried to reuse that solution! In that sense, *utility* can be considered as an *a posteriori* similarity.

The paradox is resolved by the following assumption: The more similar a previous problem is to the current problem at hand, the more useful the solution of that problem will be. Given this, similarity can be determined on the basis of the given problem descriptions and, hence, becomes an *a priori* criterion.

Note that an imprecise usage of these notions can be widely observed: Often, one speaks of similarity of cases which is a misconception because of two reasons:

- Firstly, one hardly ever wants to compare two *cases*, i.e. two problem solving episodes already stored in memory. Rather, one usually wants to compare a *query* with a *case*. This is a crucial difference as the query is not necessarily represented in the same way as a case.
- Secondly, not entire cases are compared to compute similarity but most often the *problem descriptions* of cases are — or with the previous point the problem description of a case is compared to the problem description of the query.

We will ignore these difficulties for now and be content with the more or less vague description given above. Later on, when discussing the memory model of Case Retrieval Nets, precise definitions of all these concepts will be given.

2.2.4 A General Process Model of CBR

The main activities required within a CBR system are described by using the R^4 cycle developed by Aamodt and Plaza (1994) as displayed in Figure 2.1. According to this, the general process model of CBR consists of four phases:

Retrieve is the process of accessing the most similar cases to a given problem in the case memory.

Reuse is concerned with applying the solution(s) of the retrieved case(s) to the current problem, including adaptation.

Revise is required to check the applicability and correctness of the solution derived in the previous phase; this is required because CBR, due to the inexact matching, may suggest incorrect solutions.

Retain is concerned with the incorporation of the "lessons learned" into the system; in the simplest case just the new problem together with its solution is added to the case base, but it is also possible to adjust the similarity measure or the adaptation knowledge (see the *Knowledge Container* model below).

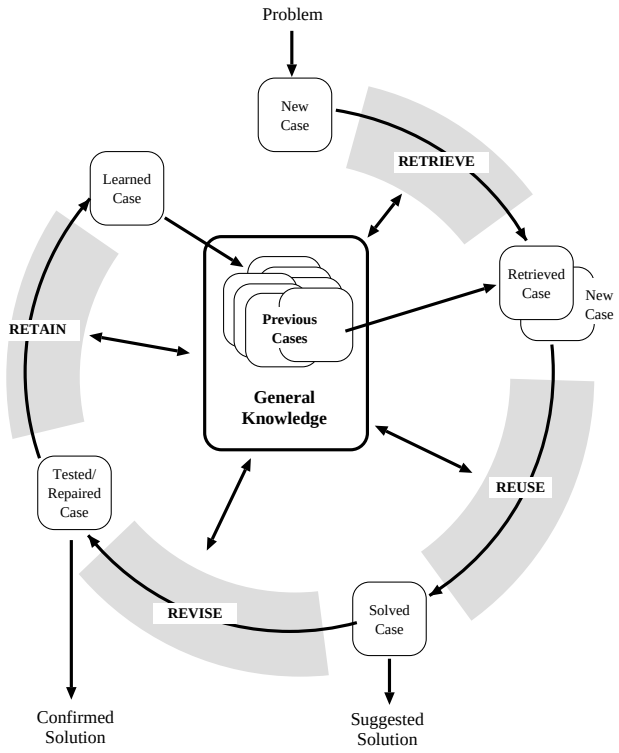


Figure 2.1: The CBR-Cycle after Aamodt and Plaza.

Each of these parts of the R^4 model describes a specific phase of the reasoning process, and each can be implemented by a number of different techniques. Retrieve, for example, includes the identification of the relevant features, the search through the case memory, the calculation of the similarity values, and the selection of the most similar cases. The search itself can again be implemented by a number of alternative techniques, such as indexing, linear search, special search structures and so on. This is known as the *task-method decomposition model* (Aamodt and Plaza 1994).

The R^4 model has now become a kind of standard at least in the European community. Of course, not every specific system has to implement all of the above four phases. Certainly, all will perform case retrieval, but it has to be decided, for example, whether automatic Reuse (adaptation) is feasible and whether there should be a Retain phase in the sense that the system *automatically* adjusts its internal structures. In certain circumstances, this might require appropriately defined management processes.

Besides this model, other authors suggest more or less different processes. For example, Allen (1994) suggests the five phases Presentation, Retrieval, Adaptation, Validation, and Update. So, apart from the Presentation phase, this model can be mapped directly to the R^4 model by Aamodt and Plaza. What is important, however, is that R^4 models the Case-Based Reasoning process as a *cycle*, i.e. it is made explicit that learning and problem solving are closely integrated. This is not made explicit, for example, in Allen's model.

2.2.5 Knowledge Containers

The concept of *knowledge containers* has been introduced by Richter (1995); it appears to be particularly useful for describing what kind of knowledge can be incorporated in a CBR system

and how these components interact (see also Richter 1998).

A knowledge container is used to take knowledge of a certain structure. For example, in traditional rule-based systems there are three containers, namely facts, rules, and an inference engine.

In Case-Based Reasoning, four major containers can be identified:

- the *vocabulary* used to describe cases;
- the *similarity measure* used to compare cases;
- the *case base*, or case memory, containing all the stored cases;
- the *adaptation knowledge* required for transferring solutions.

There is no strict separation between these four containers with respect to the overall problem solving task. That is, each container can, in principle, carry any piece of knowledge and provide this for solving the problem at hand. For example, when having a complete case base (i.e. a case base in which each potential case is stored), one would need neither a similarity measure (this could be reduced to identity) nor adaptation knowledge. In fact, the entire system could be implemented as a traditional database.

On the other hand, when having a complete model of adaptation, one could start from virtually any case and adapt it to fit the requirements of the current problem. So, no case base and, hence, no similarity measure would be required and the entire system could be seen as a model-based system.

With respect to how the above four containers are used in CBR there is a crucial difference which has severe consequences for knowledge acquisition: While the knowledge contained in three containers is used at compile (or coding) time, the *case base* is interpreted at run time: In most practical implementations of CBR systems, there is a predefined scheme of how cases may be represented; also there are functions implementing the computation of the similarity measure as well as an implemented strategy for the solution transformation. The case base, however, may be changed at any time while the system runs.

This is an important advantage of CBR as the knowledge acquisition bottleneck known from traditional expert system technology is circumvented to a large degree: It is not required that all the information which is required for running the system is available at coding time.

A further advantage derives from the fact that the four knowledge containers are, to a large degree, independent from each other: Manipulating the content of a container does have little consequences on the other ones. For example, adding new cases does not influence the similarity measure. In that sense, the knowledge stored in these four containers is easier to maintain than in traditional knowledge-based systems.

2.2.6 Types of CBR Systems

Case-Based Reasoning is concerned with the integration of problem solving and learning. Consequently, CBR systems usually are characterized according to the type of task they perform rather than according to their internal structure or technical implementation issues.

In the following, we will briefly characterize the main ideas of each task type. We will start with four tasks that belong to so-called *analytic* problem solving which is concerned with analyzing, or interpreting, a given solution and driving inferences based on these interpretations.

After that we will address configuration, planning, and design which are *synthetic* tasks in the sense that these have to compose some new piece of knowledge which has not been available before. These three tasks types are just named here to achieve a certain completeness; we will not address them in the remainder of this thesis.

Case-Based Classification

In classification processes, one deals with a universe of objects which are assigned to predefined classes. The goal is to automatically assign the appropriate class to a new object. More precisely:

Definition 2.1 (Classification) *Let U be a universe of objects and let further subsets $K_i \subseteq U, i \in I$, be classes. A* classifier *is a function*

$$\mathsf{class} : U \to I$$

such that $\mathsf{class}(x) = i$ *implies* $x \in K_i$. □

The fundamental assumption of case-based classification is that if two cases $c_1 = (x_1, \mathsf{class}(x_1))$ and $c_2 = (x_2, \mathsf{class}(x_2))$ have similar problem descriptions x_1 and x_2, then they will have similar or even the same classifications:

$$x_1 \approx x_2 \longrightarrow \mathsf{class}(x_1) \approx \mathsf{class}(x_2) \tag{2.1}$$

Given this, a *case-based classifier* is usually represented as a pair

$$(CB, \mathsf{sim})$$

where $CB \subseteq U$ is the given case base and sim is a similarity measure

$$\mathsf{sim} : U \times CB \longrightarrow [0, 1]$$

which allows to define the *nearest neighbor* $NN(x)$ for a given case x. Once the classes of all $c \in CB$ are known, the class of a new object $x \in U$ can be computed by determining the class of its nearest neighbor:

$$\mathsf{class}(x) = i \text{ if } NN(x) \in K_i$$

Due to CBR using the concept of similarity, this classification may, in fact, be incorrect in that an object is assigned a class that turns out to be false. Consequently, a case-based classifier, in general, only defines an approximate classifier appr_class where it is possible that

$$\exists x \in U : \mathsf{appr\_class}(x) = i \wedge x \notin K_i$$

A major advantage of CBR is that learning can be integrated into case-based classification and hence an initially *approximate* classifier function appr_class can be adjusted such that appr_class becomes a correct classifier over time.

Also, the fairly simple notion of the nearest neighbor may be refined by, for example, taking into account the k nearest neighbors and letting these *vote*.

At this point two remarks are due:

- Firstly, a crucial requirement of case-based classification is that the elements of the case base are encoded as pairs consisting of the actual cases and the classification, or solution of these cases. That is, the class has to be a specific, a priori defined element of a case. As we shall see in Chapter 3 this causes a number of problems in understanding more complex problem solving strategies.
- Secondly, the reverse of implication (2.1) does not hold in general: Two defendants may well receive the same sentence although they have committed completely different crimes (see Richter 1998 and Burkhard 1998a for discussions on general aspects of similarity).

Example 2.2 A typical application of case-based classification could be automatic quality control in manufacturing: A robot might monitor a production process and protocol important parameters of both the products and the machinery used (e.g., temperatures, pressures, noises, colors, etc.). At any time during the process, each produced item can be classified according to whether or not it is of sufficient quality.

Case-Based Diagnosis

While Puppe (1993) uses the terms classification and diagnosis as synonyms, the latter usually includes a *process* of ascertaining (further) symptoms to improve the quality of problem solving. In contrast, all features are typically known at the beginning of a classification process.

To emphasize this distinction, Richter (1992) introduces a third set of domain objects besides problems and solutions, namely tests which can be performed at any time during a diagnostic process in order to obtain knowledge about further symptoms.

Furthermore, while the main objective remains finding diagnoses, subtasks may aim at identifying differences between sets of observations in order to determine which test should be performed next. This is usually referred to as *differential diagnosis*.

The selection of the most appropriate tests is itself a difficult problem which may again be guided by cases. These cases are often called *strategic cases* (Althoff 1992).

Another difference to classification is that in diagnosis very often the reverse of relationship (2.1) is assumed: It is quite natural to assume that, if similar diagnoses have been established in two situations, then similar symptoms should be observable, too. Some diagnostic approaches heavily rely on this assumption, e.g. when using Bayes' theorem to derive a diagnosis for an observed set of symptoms given the knowledge about which symptoms are expected for certain diseases (Heckerman 1991; Pearl 1988).

Example 2.3 A typical case-based diagnostic system, which also implements differential diagnosis, is the Cassiopée system developed by AcknoSoft for diagnosing aircraft engines (Heider 1996).

Case-Based Decision Support

Decision support is even more general than diagnosis in the sense that there is no specially marked feature as a diagnosis which should be considered as the solution of a problem. Rather, it is often unclear in advance, which features of the domain (i.e. elements of the vocabulary) will be used for describing future problems and which will be considered as solutions.

In the context of decision support, a problem is simply a representation of a situation with missing information. The objective is to complete the description of that situation during problem solving such that a certain demand for information can be satisfied (cf. Richter 1992; Lenz, Burkhard, Pirk, Auriol, and Manago 1996; Burkhard 1998a).

Furthermore, it is usually not possible to (automatically) decide whether a solution (i.e., a piece of information) found by a decision support system (DSS) is *correct* or not. Rather, this information may be more or less useful, it may be better or worse than some other piece of information. This problem is due to the following domain characteristics:

- DSSs are most often applied in domains where the terminology is highly ambiguous.
- The interpretation of certain pieces of information is context-sensitive and depends on the intention of the human user.
- The existing domain knowledge is, in general too weak to allow for strong problem solving methods.

Problem areas in which most of these problems have to be coped with, are usually referred to as *weak-theory domains*.

In addition to these features, Richter (1992, p. 343) identifies four characteristic properties of decision support systems:

1. The amount of information that has to be coped with is too large to be handled by humans without support by an appropriate technical system.

2. Decisions have to be made rapidly.
3. Data has to be prepared in some way for decision making.
4. The process of decision making is highly complex.

Due to these properties which make it hard to automate the decision making process, a decision support is usually built in order to (as the name indicates) *support* the (human) decision makers rather than replacing them as discussed already in Section 1.1.2.

As we shall see in Chapter 3, decision support as well as classification and diagnosis can be described as specific instantiations of information completion processes.

Example 2.4 A vivid example for a decision support application is the ICONS system as described by Gierl, Bull, and Schmidt (1998): Because of the enormous complexity of the problem space in medical domains and due to the many different aspects that have to be considered, ICONS is restricted to providing information relevant for the current case. It is the human expert who finally has to come up with a decision and take the responsibility.

Case-Based Knowledge Management

A fairly new direction of research is concerned with managing the knowledge within a company or an organization. While this is primarily an issue of enterprise organization and business management, there are a lot of areas which can be supported (or even enabled) by modern information technology in general and information systems in particular (Abecker, Bernardi, Hinkelmann, Kühn, and Sintek 1998). Some of these areas appear to be particular promising areas for CBR, such as:

Knowledge Preservation: maintaining an inventory of knowledge available in the organization;

Knowledge Dissemination: bringing the knowledge existing somewhere to places where it is needed as well as providing processes for knowledge sharing.

Knowledge Reuse: assuring that knowledge is reused and not reinvented.

The key difference to diagnostic and decision support tasks probably consists in the complexity of the problem solving process as well as the integration issues that have to be addressed: While it is, in general, possible to build a diagnostic system that uses the above defined knowledge containers to answer queries and perform a diagnostic process, a huge number of different knowledge sources exist in organizations that have to be taken into account for knowledge management. It may be hard, if not impossible, to specify the appropriate knowledge containers for all these knowledge items at compile time.

It may even be the case that further knowledge containers beyond those mentioned in Section 2.2.5 are required, for example for representing business processes that have to be followed during knowledge preservation (e.g. *Who is allowed to modify the case base?*) and knowledge utilization (e.g. *Who has access to the information?*).

Also, knowledge management is not only concerned with how people and organizations *use* the system in the sense of performing a retrieval process. Rather, users also permanently change the system by adding new knowledge, modifying previous etc. In this respect, knowledge management is much more tied to the existing business processes than this is the case for diagnostic applications where a well-defined interface can be used for exchange of data.

Example 2.5 The SIMATIC KNOWLEDGE MANAGER that will be presented in Chapter 9 can be seen as a CBR system implementing a special aspect of knowledge management, namely content-oriented retrieval of *know how documents*.

Case-Based Configuration

Configuration is the construction of an artifact from known parts taking into account the known interrelationships between these parts. In contrast to design (see below), configuration can be modeled as a closed-world problem where classical AI problem solving methods are applicable. Configuration is strongly related to constraint satisfaction problems.

From a case-based reasoning perspective, configuration and adaptation are closely related: On the one hand, adapting previous solutions to new requirements can be achieved by configuring the previous cases appropriately. On the other hand, configuration is often based on reusing previous solutions and adapting them as needed.

A comprehensive model of how these two techniques interact has been developed by Wilke, Smyth, and Cunningham (1998).

Case-Based Design

Design is the construction of an artifact from parts that may be either known and given or just newly created for this particular artifact. Depending on the degree of creativity involved we distinguish between routine design, innovative design and creative design.

CBR is particularly interesting for innovative design tasks where classical AI problem solving methods are not applicable. Due to the complexity of the problems to be solved and the lack of formal methods, approaches to design (in particular to innovative and creative design) have also some characteristics of decision support systems in that they aim at *assisting* the users rather then replacing them.

A more detailed overview is given by Börner (1998) as well as by Kolodner (1993).

Case-Based Planning

Planning in general consists of the construction of some course of actions to achieve a specified set of goals, starting from an initial situation. For example, determining a sequence of actions required for transporting goods from an initial location to some specified destination is a typical planning problem.

While the classical planning process consists mainly of a search through the space of possible sets of operators to solve a given problem, new problems are solved by reusing and combining plans or portions of old plans in case-based planning. Important concepts for these tasks are the use of cases at different levels of abstraction (Bergmann and Wilke 1995; Bergmann 1996) as well as the replay of problem solving episodes as known from derivational analogy (Carbonell 1986; Veloso 1994).

A more comprehensive description of the issues related to case-based planning has been given by Veloso, Muñoz-Avila, and Bergmann (1996). Also, Bergmann, Muñoz-Avila, Veloso, and Melis (1998) give a concise overview.

Chapter 3

Problem Solving Viewed as Information Completion

Because case-based reasoning is a natural, intuitive process for people, interactive aiding systems that help a user to solve a problem work well.
J. Kolodner, Case-Based Reasoning, Morgan Kaufmann, 1993

In this chapter, we will discuss some central issues related to case representation and case retrieval. In particular, we will show that the commonly applied definition of a case being a problem plus a solution is not appropriate in certain circumstances. This will lead us to a more general view on problem solving processes.

3.1 *Case = Problem + Solution* Revisited

3.1.1 The Traditional View on Cases

From a cognitive perspective, a case is simply a record of an experienced problem solving episode. When it comes to the implementation of practical systems, however, a more precise definition is required. For this, we presented the widely adopted notion of a case consisting of a problem description and a solution in Section 2.2. Underlying this interpretation is the assumption that every case can be divided *a priori* into these two parts.

To further illustrate this approach consider the typical definition of a classification process. For example, Puppe (1993) describes classification as a

> *"... problem solving type in which the solution is selected from a set of given solutions, ..."*

More precisely, the following properties are usually assumed for classification:

1. The problem domain consists of two disjoint sets of domain objects, namely
   - a set of symptoms (or observations) and
   - a set of problem solutions (or classes).
2. A problem description is represented as a set of observations.

3. The solution of a problem (and thus the result of classification) is the selection of one or more classes.

This kind of separation between symptoms and solutions has been widely adopted for the design of Case-Based Reasoning systems, too. What is crucial here is that a clear distinction between symptoms on the one hand and solutions on the other hand is assumed.

While CBR has been applied to a number of tasks beyond classification, the assumption of disjoint sets of symptoms and solutions often remains: More generally speaking, CBR is considered as a problem solving method which aims at finding a suitable solution for a given problem description. Consequently, it seems only natural to represent cases as pairs in the above sense.

3.1.2 Consequences for Applications of CBR

On first sight, the above described case scheme seems adequate but, in fact, this kind of case representation has a number of severe consequences with respect to

- the techniques that can be used to implement a specific CBR system;
- the reuse of cases for different purposes;
- the modeling of problem solving *processes*.

Applicable Techniques

A number of specific techniques rely on a clear separation of problems and solutions. For example, when applying a retrieval strategy based on some kind of decision tree or clustering, a kind of *class* is required for building the tree — most often this is the solution of a case.

Flexible Reuse of Cases

A major advantage of CBR is that cases can be reused in different situations and for different purposes. Examples of people using different *views* on cases can be observed in every day life:

- A customer asks the travel agent for a package that suits his needs for specific leisure activities; a second customer has limitations with respect to the available budget. Both could be satisfied by the same case, i.e. the same tour package. In fact, here the entire case comprises the solution.
- The argumentation of the attorney and the defense counsel in favor respectively against a defendant may be based on the same precedents.

For certain tasks, this kind of *flexibility* would also be beneficial for Case-Based Reasoning systems. If, however, a case has predefined slots for the problem description and the solution, this might be hard if not impossible to achieve.

> **Example 3.1** Imagine a diagnostic application where the description slots of cases would, naturally, contain the observed sets of symptoms while the solution slots would take the diagnoses. Due to the traditional distinction between problems and solutions, it would not be possible to query the system for the symptoms usually connected with a given diagnosis, for example for confirming or rejecting a certain hypothesis.

Representation of Problem Solving Processes

Most often, problem solving is not performed in a single step in real life. Rather, a *process* of analyzing the problem situation, evaluating possible hypotheses, ascertaining further features, again evaluating possible hypotheses etc. is performed until a satisfactory result is achieved. If one would like to implement this more general view on problem solving with the above assumption of cases being separated in problems and solutions, each case had to be present in case memory more than once: Several distinct *instances* of the case had to serve the purpose of ascertaining features while yet another *instance* had to represent the final solution.

As an example, consider a situation in which one wants to diagnose why a `light bulb does not work`. Similar cases may suggest that either the `switch is off` or there is `missing power`. After having checked that the `switch is on` but there is no power, one would add `missing power` as another feature and obtain cases suggesting `cable broken` and `fuse broken`, respectively. Assuming that these explanations provide sufficient granularity, one of these two solutions could be accepted as the final results of the diagnostic process. This could be sketched as follows:

PS cycle		Problem description	Solution
1	$query_1$	`light bulb does not work`	?
	$case_1$	`light bulb does not work`	`switch off`
	$case_2$	`light bulb does not work`	`missing power`
	$case_3$	`light bulb does not work`	`missing power`
2	$query_2$	`light bulb does not work` `missing power`	?
	$case_4$	`light bulb does not work` `missing power`	`cable broken`
	$case_5$	`light bulb does not work` `missing power`	`fuse broken`
3	*solution*	`light bulb does not work` `missing power`	`fuse broken`

Furthermore, $case_4$ might refer to the same problem solving episode as $case_2$ and $case_5$ might correspond to $case_3$. So, each of these would, in fact, have to be presented in case memory multiple times.

3.1.3 Consequences for Decision Support Processes

As discussed above, the flexible reuse of cases as well as the ability to model problem solving processes are crucial in particular if we want to address issues of decision support which, as discussed in Section 2.2.6, has to cope with ambiguous terms, vague and maybe inconsistent problem descriptions in highly complex domains.

When building such a system, the requirements on the system are weakened in the sense that an *automatic* problem solving capability is no longer the (unrealistic) ultimate goal. Rather, it is sufficient if, during an *interactive* process, a (human) user is *supported* in making better decisions.

For example, obtaining information about previous events can be extremely helpful for the user. Even if this information is provided *as it is*, i.e., without a further inference process, it may well give hints for the current problem at hand, such as ways to overcome the problem, obstacles to avoid, or undesired side effects.

However, it is not sufficient to recall just the outcome of the previous events (in the sense of classification) because this may hide the specific conditions that caused the outcome. Rather, the complete description of the episodes has to be available in order to allow for a detailed evaluation of applicability.

Thus, the question arises as to how cases should be represented to support these problem solving *processes*. In the following sections, the concept of *information completion* will be introduced which is more flexible with respect to the tasks that can be performed with the cases.

3.2 Representation for Information Completion

Information completion is a general view on problem solving processes by means of CBR. The idea here is that each case is just some kind of collection of information. Likewise, a new case (or query) is a collection of information which, however, may be incomplete and vague. By collecting new pieces of information, the query is subsequently completed until a satisfactory level is reached which allows for a successful treatment of the overall task.

3.2.1 Information Entities

Instead of considering a case as

$$case = problem + solution$$

we will use the notion of *information entities* for representing cases.

Definition 3.1 (Information Entity) *An* Information Entity *(IE) is an atomic knowledge item in the domain, i.e., an IE represents the lowest granularity of knowledge representation for cases and queries, respectively.* □

For example, an IE may represent a particular attribute-value pair. Thus, the set of all IEs corresponds to the vocabulary used to represent cases in the particular application domain but only those terms are considered to be IEs which could not normally be further divided without loosing or changing the semantics.

Following Meadow (1991, p.41), information representations should, in general, be selected such that they support the following tasks:

Differentiation between Entities: The discriminating power of each particular element of the representation should be maximal, that is it should apply only to one object in the entire domain. Whenever such a *code* is mentioned, it should ideally allow to identify a single object.

Accurate Description: Descriptiveness is related to uniqueness: Although codes should be chosen such that they clearly distinguish the objects in the real world, each particular code should hold sufficient information about such an object.

Minimizing Ambiguity: Meanings should be as unique and unambiguous as possible. This is related to descriptiveness but puts particular emphasis on differentiating objects rather than on completeness of descriptions.

Identification of Similarity: Apart from describing single domain objects, an appropriate representation should also contain information about similarity and dissimilarity of such objects.

Often, the above goals conflict with one another and the appropriate choice may even depend on the application and the specific type of task in mind.

Example 3.2 When wanting to retrieve information from natural language texts, one might consider using all letters of the alphabet as elements of the information representation. Then, the discriminating power will be maximal, but each letter will not really describe a single object in the domain. Also, representing knowledge about similarities (such as synonyms) will be hard.
However, if the task is to build a spell checker, the letter–based representation might be fully appropriate.

We shall see in the course of this thesis, that IEs should primarily be chosen such that the first two criteria are satisfied in the given domain. Similarity relationships will be represented by an additional element of the information representation. Ambiguity is hopefully avoided by focusing on a special domain and task that an information system has to deal with.

Note that for a given case base, the set of all IEs may change depending on the application scenario: While for the user of some technical equipment it may suffice to diagnose that a certain component is broken, the technician responsible for repairing this component may require much more detailed information.

3.2.2 Cases and Queries as Sets of IEs

Given the above notion of IEs, we may define:

Definition 3.2 (Case and Query) *A* case *consists of a unique case descriptor and a set of IEs. Similarly, a* query *is just a set of IEs.* □

This kind of representation of cases and queries has a number of specific properties:

1. The number of IEs contained in a case or in a query is not fixed but depends on the amount of information given in that particular situation.
2. As the definition explicitly specifies cases and queries to be *sets* of IEs, there will be no element occurring more than once.
3. Also, there is no direct relationship between the IEs belonging to a case or a query. Such relationships might be represented, however, by additional IEs which encode this relation (see below).

3.2.3 Types of IEs

For real-world applications, information entities may be of various types. In particular, cases are often represented by means of vectors consisting of attribute-value pairs for a finite number of given attributes. This induces a structuring of the set E of all information entities into disjoint sets E_{A_i} where each E_{A_i} would contain all attribute-value pairs in E that correspond to the attribute A_i.

This kind of segmentation of E has two major consequences:

- Firstly, if cases are attribute-value vectors each case/query may have at most one value per attribute, that is in the representation of each case at most one value from each E_{A_i} could be present.
- Secondly, as we will compare different IEs using a similarity measure later on, it seems reasonable to remark that usually values belonging to different attributes are *incomparable*, that is they will have zero similarity. However, according to the above definitions this is not required.

In the following chapters, we will use the expression $type(e)$ to refer to the attribute of which the IE e represents a certain value:

$$type(e) = A_i \text{ if } e \in E \text{ represents a value of attribute } A_i$$

3.2.4 Representation of Relationships by IEs

As mentioned above, information entities may be used to explicitly express relationships between other IEs. By doing so, *structural similarity* could be considered which, in certain domains, may be more helpful than *surface similarity* (Gentner 1983; Börner 1998).

As long as there is a finite number of objects and possible relations between these objects, each relationship that occurs in the case base can be described by a specific IE. For example,

`_IE_overlap_event1_event2_`

could be used to represent that two objects in the domain represented by IEs, namely `event1` and `event2` overlap.

In order to assess structural similarities, as for example the relation between

`_IE_overlap_event1_event2_`

and

`_IE_overlap_event3_event4_`

the relationships between the IEs have to be considered. That is, when analyzing the case data, these relationships must be contained in one of the knowledge containers provided for the entire system. Hence, they could, in principle, be compiled to be explicitly present.

Nevertheless, we realize that this kind of approach is probably not best suited for domains with highly structured case data, such as the FABEL domain of industrial building design (Gebhardt, Voß, Gräther, and Schmidt-Belz 1997; Börner 1998). However, the main focus when developing the models presented in this thesis has been on application domains with a somewhat simpler case format where other problems have to be addressed.

3.3 Information Completion Processes

In this section, we will give a schematic description of how a case retrieval process could be implemented which is based on the more flexible representation of cases and queries by sets of IEs.

The idea is straightforward: Given a query, the case memory is searched for similar cases – or, more precisely, for cases having a similar description[1]. Components of these similar cases are candidates for completing the information contained in the query, i.e., they have to be checked for consistency with the already known pieces of information and can then be added to the query description.

Alternatively, the additional information may not come directly from the most similar cases but these might suggest tests to be performed in order to obtain new information from the *real world*; Burkhard (1998a) called this *case completion* then. In any case, information completion heavily relies on the retrieval phase of Case-Based Reasoning during which a preference ordering of cases is established which will guide the process of information completion.

In contrast to, for example, classification tasks, a formal definition of information completion processes is not straightforward. Rather, a number of domain- and application-specific criteria have to be considered, such as:

1. What similarity function will be used for establishing the preference ordering?
2. Which cases will be selected: Does it make sense to accept the k most similar ones? Should exceptional cases be preferred, or a set of cases be selected which differ concerning important aspects?

[1] We adopt here the common notion of *similarity*, although Burkhard (1998a) clarified some misconceptions in particular when considering similarity as a kind of inverse distance. To be more precise, we should use *acceptance* instead.

3. What does it mean to *complete* missing information? Can it be extracted directly from the retrieved cases or do these cases suggest tests to perform for obtaining new insights (such as in the process of *differential diagnosis*)?

4. When will the desire for more information be satisfied, i.e., when can the problem initially represented by the query be considered as being solved?

In particular, property 3 makes it impossible to give a general description of an information completion process in an algorithmic manner as the information that is obtained in this step is not necessarily represented within the system but may be the result of a test procedure performed in the outside world or a subjective decision of the (human) user.

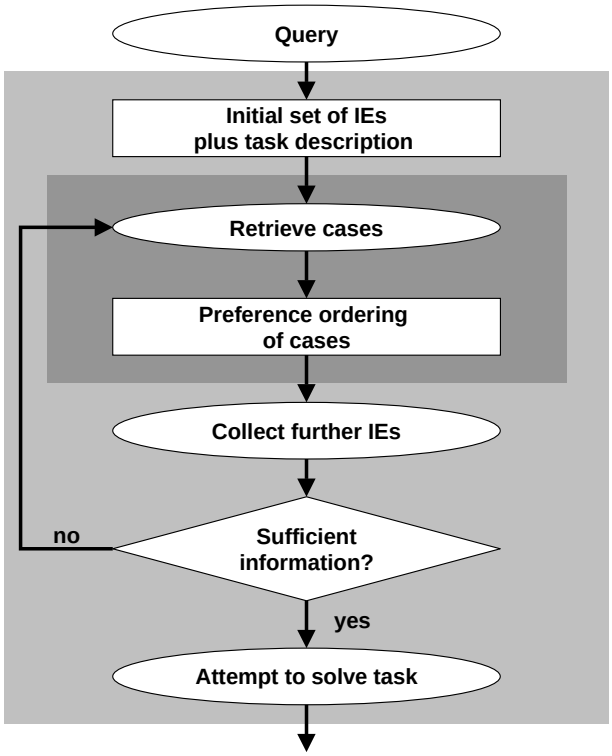


Figure 3.1: Schematic view on information completion

Consequently, we will describe information completion by means of a less formal process diagram as shown in Figure 3.1. According to this schematic view, four sub-processes of information completion can be distinguished:

1. Posing the query to the system

2. Retrieving cases based on the information collected so far

3. Collection of further pieces of information

4. Attempting to solve the problem

Even when finished successfully, the result of this process is not really a *solution* to the initial problem which may readily be applied to solve it. Rather, in a decision support setting the retrieved information can be used by a (human) expert in the decision making process.

The actual implementation of the required processes gives rise to a number of different approaches to information completion:

- Traditionally, retrieval is performed based on the query only and during collection of information just the most similar cases to that query are accessed.
- In contrast, the model of OCRNs applies the above scheme in that cases are used to collect further IEs (cf. Section 6.3).
- The collection of further IEs may be performed within the CBR system, in cooperation with some other external system, or by means of asking the user.
- Likewise, the decision about whether sufficient information has been gathered or not and the actual problem solving step may take place either within the CBR system or externally.

Note that classification and diagnosis as defined in Section 2.2.6 are just special cases of information completion: For classification tasks, the symptoms are used for querying the case base, and the found class(es) complete the query. Additionally, tests for further symptoms can be derived for diagnostic tasks.

Part II

Theory of Case Retrieval Nets

Chapter 4

Starting Points for Case Retrieval Nets

... memories can be brought back into working memory by a spreading activation process by which associated concepts retrieve ... facts
John R. Anderson, A Theory of the Origins of Human Knowledge, 1989

After having outlined the idea of considering problem solving as information completion, we will in the following chapters give a precise definition of a memory model that is well suited for this task. In this chapter, we will explain what the starting points in the development of this model had been. In particular, we will address issues of representation and of the retrieval process.

4.1 Distributed Representations

An important property of human memory is that it is content-addressable. That is, remindings can be retrieved from memory for virtually any given retrieval cue. There is no pre-selected index which is primarily used for retrieval purposes. Of course, the quality of the cues may differ in the sense that some features are more effective than others in that they more accurately help to pick out the desired memory representations. Also, a single cue will rarely ever be discriminant enough for identifying the relevant information. But several such features, in conjunction, most often will be sufficient.

A major advantage of content-addressable memories is that they are less vulnerable to errors and missing information. If, for example, a set of retrieval cues is given and one of these cues is wrong, then most traditional case-based memories would not be able to deliver the appropriate cases. A decision tree–based retrieval, for instance, could be misguided by the wrong cue even on one of the top levels of the tree and, thus, could be directed into completely wrong regions of the case memory. The situation is different for content-addressable memories where the targeted item in memory (i.e. the most similar case) would still be reached via all the other retrieval cues.

A straightforward way to implement such a content-addressable is to consider both the items in memory (that is, the cases) as well as the potential cues (i.e. the features of the cases) as nodes in a net and to insert arcs between a *feature node* and an *item node* if the particular feature could be observed for that item.

Figure 4.1 illustrates this for a simple domain where data about a number of people has to be stored in memory, such as the name, the approximate age, and whether they are single, married, or divorced. In that model, each of the black nodes in the center of the figure represents a particular person, the other nodes encode specific features, and it is possible to access a person's record from each of the represented cues in memory.

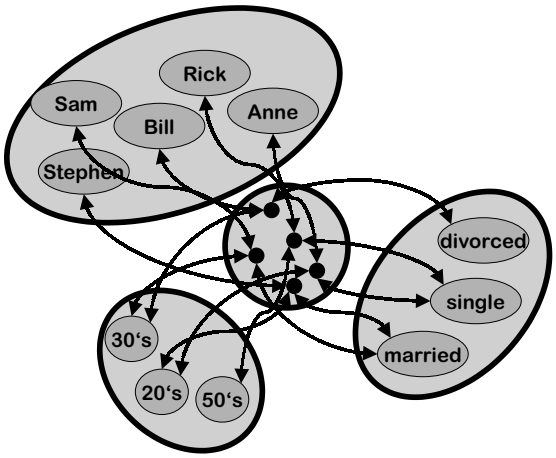


Figure 4.1: A content-addressable memory for storing people's data (adapted from Rumelhart, McClelland, and the PDP Research Group 1986).

The model depicted in Figure 4.1 makes use of a so-called distributed representation. That means that the memory objects (i.e. the people's records) are not represented in a monolithic way but rather may share memory structures if the same feature can be observed. For example, `Sam` and `Bill` are both in their `30's` but this feature is represented in memory just once.

This type of model already serves as a good starting point when thinking about how a case memory might look like that is similar to content-addressable memories. Of course, what is missing here are the relationships among the various features, i.e. similarities among the different ages or family statuses. We shall see in Chapter 5 how this can be incorporated in a similar model.

4.2 Retrieval without Search?

In Section 3.1 we discussed the limitations of a more or less widely applied scheme of case representation. After having presented the idea of using the more flexible concept of information entities, the question arises how the required Case-Based Reasoning processes have to be designed in order to be able to handle this type of structuring. In particular, we will consider here the Retrieve task.

To illustrate the consequences of using information entities for the retrieval process, we will discuss in the following some important issues related to case retrieval techniques.

4.2.1 Remindings: Search versus Reconstruction

A widely accepted idea in the CBR community is that case retrieval has to be performed by some kind of *search* process through the case memory. Even more, problem solving in Artificial Intelligence, in general, is strongly tied to methods of searching a structured memory. This dates back to the early 1970's when Newell and Simon (1972) suggested to consider human problem solving as a search process from a cognitive point of view.

However, there is growing evidence that humans are able to solve problems *without* performing an intensive search process. For example, Enzinger, Puppe, and Strube (1994) argue why a search process in the traditional sense is neither possible nor necessary in human problem solving. One of the major arguments is that in particular experts invest much more time in *analyzing* problem situations than novices do. Due to this, they are able to come up with solutions faster *despite* the fact that they have more experiences (and hence the search space is larger).

Another idea relevant for case retrieval has already been explored by Bartlett (1932) who suggested that remembering should be considered as a *reconstruction* rather than a search process, i.e. humans remember things by starting with some initial clue and subsequently collecting more and more pieces of information until the *picture* is sufficiently complete.

In fact, these ideas can also be found in the dynamic memory model of early CBR systems (Schank 1982; Kolodner 1983b). More precisely, the following principles are claimed to be essential for a true model of human learning and problem solving:

1. The structures in memory that are used for processing (i.e., the ones that provide expectations and suggest inferences) are the same ones that are used for storage.

2. Reminding in human beings is reconstructive. That is, most often an event is remembered according to the expectations of that event instead of a pure recall.

3. Rather than directly enumerating items in memory, descriptions of episodes are constructed and used for accessing the entire episode.

4. The process of reminding consists of progressively narrowing in on a description of the item to be remembered.

5. In memory, similar data is maintained by means of their differences via a *category* which corresponds to their agreement.

Thagard and Holyoak (1989) present an implemented system, the ARCS[1] system for performing retrieval of analogs, which implements the idea of *constructing* a previous situation rather than *searching* for it. They emphasize:

> *"It is important to be clear about what* ARCS *is* not *doing: It does not compare the probe with every structure in memory, but considers only those that have semantically similar predicates. Nor does it do a complete match between the probe and the source analogs whose potential relevance is indicated by semantic similarity."*

4.2.2 Case Retrieval by Searching

As case bases grew and systems became more application-oriented, a tendency towards *top-down* search methods could be observed, such as kd–trees (Wess 1995). To illustrate the consequences of this, let us consider the following comparison which will show major differences with respect to the principle behavior of the techniques:

When similar cases are desired, one usually accepts a search in the sense that at the beginning of this search all cases are considered potential candidates and this set is subsequently restricted. If, however, humans search for a similar situation, they often start from the given description, consider the *neighborhood*, and extend the scope of considered objects if required.

[1] Analog Retrieval by Constraint Satisfaction

Example 4.1 If, for example, the task is to find the city closest to `Cologne`, a natural approach is to find `Cologne` on the map and to look in its neighborhood for appropriate candidates. If none is found in the direct surroundings of `Cologne`, then the scope of search is subsequently extended. If two or more are found from which no single best candidate can be selected right at first glance, then the actual distances are computed, compared, and the closest city can be retrieved.

In AI terminology, the search techniques usually applied in the CBR Retrieve phase can be considered as *top-down* processes in the above described sense of subsequently restricting the set of potential candidates. Russel and Norvig (1995, p.70) describe this as follows:

> *"...finding a solution ...is done by a search through the state space. The idea is to maintain and extend a set of partial solution sequences."*

Similarly, Kaindl (1989) describes a range of possible problem solving techniques typically used in Artificial Intelligence. Some of these techniques use specific heuristics to prune the search space, others don't. But all are based on what is called here *top-down* search methods.

In contrast to this, humans obviously perform a *bottom-up* process in certain situations: They analyze the problem, derive features specific to that problem, and use these features as direct links to memory

Of course, one might argue that human reasoning is based on a much more elaborated memory structure which may also perform parallel reasoning processes. As long as such techniques are not available at computers, *search is the only way of doing the job.* However, this is only partially true. There are at least two techniques which are capable of performing problem solving in a *bottom-up* fashion, namely neural networks and content-addressable memory.

If, for example, connectionist methods are used for problem solving, than the description of the problem is used to initially activate appropriate structures and by means of some internal processes a solution is provided by the system. A search in the sense of AI does not take place. (As a consequence, the number of data items stored, e.g. in a Neural network does not directly influence the time required for problem solving.)

For CBR, similar solutions might be desirable, i.e. retrieving a case because the components of its problem description *directly* suggest that this case might be useful (cf. the ideas of *hooks* attached to cases according to the case descriptions by Aamodt 1991).

Note the difference here to the usual understanding of *indexing* techniques in CBR: Indexes normally are predefined memory structures which are assigned to cases in order to access them later on. Instead, neural networks directly utilize the descriptions of problems as they are — no additional indexes are present.

4.2.3 Case Retrieval by Association

An idea that strongly influenced the development of the Case Retrieval Net models, is that the case memory can be organized as an associative memory and case retrieval should be performed within these memory structures similar to the processes known from both connectionist models and associative memory models.

The underlying assumption is that if a case is similar to a query, than both will share some properties in the sense that they are described by similar features. Consequently, if a query is submitted to such a system then the features present in the description of the query can be used to directly access the cases that share the same features. In order to also have access to cases sharing not the same but similar features, additional memory structures are required which represent the similarity relationships between different features. This is exactly what the associative structures are used for.

Of course, Case-Based Reasoning differs significantly from both, neural networks and associative memory models. Nevertheless, it is possible to borrow some of the ideas developed

in these communities to come up with a specific memory model for case retrieval: Instead of building, for example, a tree from the case base, a properly structured net might be constructed. In contrast to neural networks, however, all the nodes and arcs in that net should have a precise meaning, i.e. they should be interpretable in terms of the considered application domain. Given this kind of net, one can apply a spreading activation process in order to retrieve cases being similar to a posed query. This idea will be elaborated in the following chapter.

4.3 Objects, Cases, and Indexes

A common understanding in CBR is that cases directly correspond to the objects in the real world and that these objects have to be kept in a separate case base. Given this, the Retrieve process is performed by somehow searching the case base for the most similar cases to a given query.

For the understanding of the model of Case Retrieval Nets that will be explained in subsequent chapters it appears to be useful to be more precise and to distinguish between the objects that exist in the real world, the cases that represent these objects within a CBR system, and the structures used for the Retrieve process:

Domain objects refer to the *raw* data as it is actually present in the application domain, such as a relational database or a collection of documents.

Cases are representations of domain objects internally to the CBR system.

Case descriptors allow to identify a case within case memory, such as an identifier or a primary key in databases.

Case indexes refer to those structures that are being used within the CBR system in order to perform a case retrieval process.

In general, the internal structures may be obtained from the original domain objects either automatically, or by means of a knowledge engineering process, or by a combination of both. For our purposes, we will in addition assume that

- a case represents exactly one domain object;
- a case descriptor uniquely corresponds to a case and, via the previous point, to a domain object.

Given these two assumptions, we can rely on an unequivocal relationship between domain objects, cases, and case descriptors.

Each of the processes within a case-based information system, in particular those discussed in Section 7.3, will be based upon one or more of these objects: Domain analysis and maintenance will be concerned with converting domain objects to cases, a Case Retrieval Net will be built from a set of cases, and retrieval will be performed based on case indexes and case descriptors.

Consequently, when describing both the memory structures as well as the processes within a Case Retrieval Net it would be required to explicitly state whether, for example, cases, case descriptors, or case indexes are considered. To provide a consistent way of referring to these objects we will throughout subsequent chapters write cases as $\hat{c}$ and $\hat{c}_1, \ldots, \hat{c}_n$ whereas case nodes will be written as c and $c_1, \ldots, c_n$. Likewise, the set of all cases and case nodes will be referred to as $\hat{C}$ and C, respectively.

To ease the understanding we will, however, often utilize the above mentioned unequivocal mapping between the concerned objects and omit this distinction if it is clear from the context.

Chapter 5

Basic Case Retrieval Nets

Knowledge is of two kinds. We know a subject ourselves, or we know where we can find information upon it.
Samuel Johnson, 1775

In this chapter, we will present in detail the model of Case Retrieval Nets, including a formal definition of the model, a description of the retrieval process, and an investigation of the properties of this model. The model of a Basic Case Retrieval Net *as defined in this chapter will form the base for all extended models and applications in the following chapters.*

5.1 Basic Ideas of Case Retrieval Nets

The problems discussed in Chapter 3 served as a starting point in the development of Case Retrieval Nets (CRNs). In particular, the idea of a neural network or an associative memory being used for a problem solving task without performing a search in the traditional sense of Artificial Intelligence serves as a good illustration of the type of memory which we have in mind.

5.1.1 Bits and Pieces

The most fundamental item in the context of CRNs are information entities (IEs) as introduced in Section 3.2.1. Recall that these represent knowledge on the lowest level of granularity appropriate for the specific application domain.

A *case* then consists of a set of such IEs, and the *case memory* is a net with nodes for the IEs observed in the domain and additional nodes denoting the particular cases. IE nodes may be connected by *similarity arcs*, and a case node is reachable from its constituting IE nodes via *relevance arcs*. Different degrees of similarity and relevance may be expressed by varying arc weights.

Given this structure, case retrieval is performed by

1. activating the IEs given in the query,
2. propagating activation according to similarity through the net of IEs,
3. and collecting the achieved activation in the associated case nodes.

5.1.2 An Illustration

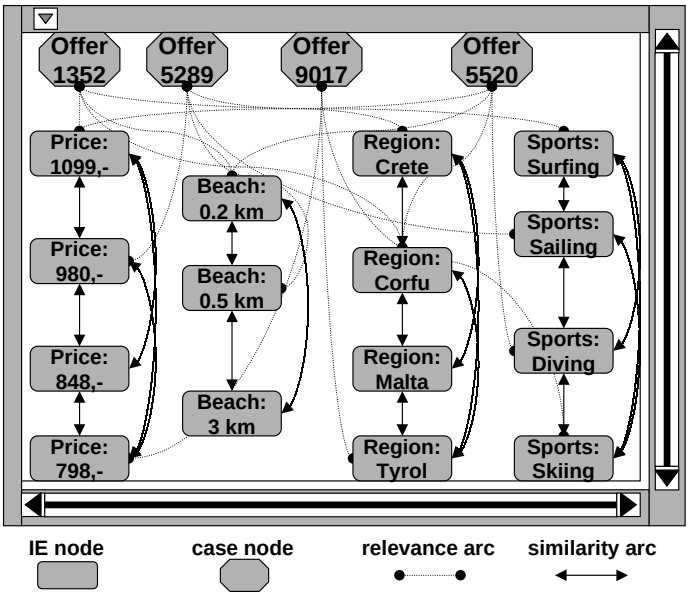


Figure 5.1: Example of a CRN in the TRAVEL AGENCY domain: IEs represent features of tour packages while cases refer to available offers. Some IEs are linked via similarity arcs, and relevance arcs exist from IEs to their associated cases.

The idea is illustrated for the TRAVEL AGENCY domain (cf. Chapter 8 as well as Lenz 1994a; Lenz 1996b) in Figure 5.1: A *case* is a special travel offer, denoted by a case descriptor, e.g. `<Offer 20219>`. It consists of a set of corresponding IEs giving the specification of that offer, in case of `<Offer 20219>` the IE nodes `<Type:Swimming>`, `<Price:980>`, `<Place:Matala>`, `<Region:Crete>`, `<Distance to beach:500 m>` are associated to that case node. Asking for an offer in Crete for swimming and not too far from the beach, the IE nodes `<Type:Swimming>`, `<Distance to beach:200 m>` and `<Region:Crete>` are initially activated. By similarity, the IE nodes `<Region:Malta>` and `<Distance to beach:500 m>` will be activated in the next step, but the amount of activation depends on the arc weights. Finally, the three offers `<Offer 20024>`, `<Offer 20219>`, `<Offer 500122>` will each get some activation depending on the incoming activations of IE nodes and their relevances. The highest activated cases may be collected and proposed to the customer. A first list of proposals might include alternative solutions which are pruned after the customer decided for either of them.

Note that in the TRAVEL AGENCY domain the traditional approach of representing a case as a problem description plus a solution would not seem adequate as it seems hard (if not impossible) to separate the items of a tour packages accordingly. Consequently, the more flexible way of using information entities as introduced in Section 3.2.1 is essential.

Furthermore, the example already shows why any two cases (or, to be more precise, a case and a query) should be considered as similar: because they are described by sets of similar IEs. If, on the other hand, two cases are highly dissimilar with respect to all IEs, then it does not seem reasonable to assume similarity on the case level.

5.2 Formal Model of Basic Case Retrieval Nets

In this section we will give a formal description of Case Retrieval Nets in a basic (*flat*) version allowing for a detailed investigation of the approach. The model will be based upon the notion of IEs as they have been specified in Definition 3.1. Also, cases are considered to be sets of IEs according to Definition 3.2. However, we will from now on assume that only those parts of cases have been represented as IEs which should be used for retrieval purposes and can, thus, be seen as a kind of index.

5.2.1 The BCRN Model

Definition 5.1 (Basic Case Retrieval Net) *A* Basic Case Retrieval Net *(BCRN) is defined as a structure* $N = [E, C, \sigma, \rho, \Pi]$ *with*

E *is the finite set of* IE nodes*;*

C *is the finite set of* case nodes*;*

σ *is the* similarity function

$$\sigma : E \times E \to \mathcal{R}$$

which describes the similarity $\sigma(e_i, e_j)$ *between IEs* e_i*,* e_j*;*

ρ *is the* relevance function

$$\rho : E \times C \to \mathcal{R}$$

which describes the relevance $\rho(e, c)$ *of the IE* e *to the case node* c*;*

Π *is the set of* propagation functions

$$\pi_n : \mathcal{R}^E \to \mathcal{R}.$$

for each node $n \in E \cup C$*.* □

The graphical description (cf. Figure 5.1) is given by a graph with nodes $E \cup C$ and directed arcs between them. The arc from $e_i \in E$ to $e_j \in E$ is labeled by $\sigma(e_i, e_j)$, the arc from $e \in E$ to $c \in C$ is labeled by $\rho(e, c)$ (arcs are omitted if they are labeled with zero). The functions π_n are annotations to the nodes n.

An IE e belongs to a case c (is *associated* to it) if $\rho(e, c) \neq 0$. Its relevance for case c is given by the value of $\rho(e, c)$ expressing the importance for re-finding e in a retrieved case c. Similarity between IEs e_i, e_j is measured by $\sigma(e_i, e_j)$. The functions π_n are used to compute the new activation of node n depending on the incoming activations (a simple setting may use the sum of inputs as π_n).

5.2.2 Similarity Propagation

Definition 5.2 (Activation of a BCRN) *An* activation *of a BCRN* $N = [E, C, \sigma, \rho, \Pi]$ *is a function*

$$\alpha : E \cup C \to \mathcal{R}$$

□

In the graphical notation, the activations $\alpha(n)$ are further annotations to the nodes $n \in E \cup C$. Informally, the activation $\alpha(e)$ of an IE e expresses the importance of that IE to the actual problem. The influence of an IE on case retrieval depends on that value and its relevances $\rho(e, c)$ for the cases c. Negative values can be used as an indicator for the rejection of cases containing that IE.

Formally, the propagation process for the basic model is given by:

Definition 5.3 (Propagation process in a BCRN) *Consider a BCRN* $N = [E, C, \sigma, \rho, \Pi]$ *with* $E = \{e_1, \ldots, e_s\}$ *and let* $\alpha_t : E \cup C \to \mathcal{R}$ *be the activation at time* t. *The* activation *of IE nodes* $e \in E$ *at time* $t+1$ *is given by*

$$\alpha_{t+1}(e) = \pi_e(\sigma(e_1, e) \cdot \alpha_t(e_1), \ldots, \sigma(e_s, e) \cdot \alpha_t(e_s)),$$

and the activation *of case nodes* $c \in C$ *at time* $t+1$ *is given by*

$$\alpha_{t+1}(c) = \pi_c(\rho(e_1, c) \cdot \alpha_t(e_1), \ldots, \rho(e_s, c) \cdot \alpha_t(e_s))$$

□

To pose a query, the activation of all IE nodes may start with

$$\alpha_0(e) = \begin{cases} 1 & : \quad \text{for the IE nodes } e \text{ describing the query} \\ 0 & : \quad \text{else} \end{cases}$$

For more subtle queries, α_0 might assign different weights to special IE nodes, and some *context* may be set as initial activation for further nodes.

Given α_0 and Definition 5.3, it is well-defined how the activation of each node $n \in C \cup E$ has to be computed at any time. In particular, case retrieval by propagation of activations is a three-step process:

Step 1 – Initial Activation:
Given the query, α_0 is determined for all IE nodes.

Step 2 – Similarity Propagation:
The activation α_0 is propagated to all IEs $e \in E$:

$$\alpha_1(e) = \pi_e(\sigma(e_1, e) \cdot \alpha_0(e_1), \ldots, \sigma(e_s, e) \cdot \alpha_0(e_s)) \tag{5.1}$$

Step 3 – Relevance Propagation:
The result of step 2 is propagated to the case nodes $c \in C$:

$$\alpha_2(c) = \pi_c(\rho(e_1, c) \cdot \alpha_1(e_1), \ldots, \rho(e_s, c) \cdot \alpha_1(e_s)) \tag{5.2}$$

Putting all the formulae together, we obtain the following result:

Theorem 5.1 *Consider a BCRN* $N = [E, C, \sigma, \rho, \Pi]$ *with the activation function* α_t *as defined in Definitions 5.2 and 5.3. The result of case retrieval for a given query activation* α_0 *is the* preference ordering *of cases according to decreasing activations* $\alpha_2(c)$ *of case nodes* $c \in C$.

5.3 Advantages of BCRNs

After the formal model of Basic Case Retrieval Nets has been introduced in the previous section, we will investigate the properties of this memory model now. In particular, we will address issues of completeness, correctness and, last but not least, efficiency and flexibility.

5.3.1 Completeness and Correctness

As discussed in Section 4.2, case retrieval traditionally is considered as a search process through case memory. In order to speed up this process, certain heuristics are often applied. Consequently, the question arises as to how the quality of the retrieval process is influenced by these heuristics, that is

- are the retrieved cases the most similar ones (correctness) and
- are the most similar cases retrieved (completeness).

As shown by Wess (1995, p. 161), three different levels of similarity have to be considered when dealing with these questions:

SIM_{App}**:** the *application* level, i.e. the similarity relationships as they are intended by some user based on his/her view of the real world;

SIM_{Rep}**:** the *representation* level, i.e. the similarity relationships as they are represented in the CBR system;

SIM_{Imp}**:** the *implementation* level, i.e. the similarity as operationalized in the retrieval procedure allowing for an efficient case retrieval.

In the current context, SIM_{Imp} could also be written as SIM_{BCRN} as it is the BCRN which operationalizes the represented similarity measure, i.e. performs retrieval.

Given this, a retrieval procedure is defined to be both complete and correct if and only if the preference orderings induced by SIM_{Rep} and SIM_{Imp} are identical. Note that this is not necessarily the case, for example when a heuristic hill climbing search prunes some parts of the case memory in order to improve efficiency of the retrieval process.

We will not go into detail about the nature of preference orderings here as this would require repeating many details already worked out by Wess (1995, Chapter 7). Instead, we will show in the following that for certain classes of similarity measures an even stronger condition is satisfied by case retrieval in BCRNs, namely that for each query the degree of similarity of each case computed by the BCRN is exactly the same as given by a similarity function on the representation level, or in slogan form:

$$SIM_{Rep} = SIM_{Imp}$$

This, of course, implies the identity of the induced preference orderings and, hence, completeness and correctness of case retrieval in BCRNs. More precisely, we will show that a BCRN can actually *implement* the computation of a certain class of similarity measures:

Definition 5.4 (Similarity implemented in a BCRN) *For a given case base $\hat{C}$, a BCRN $N = [E, C, \sigma, \rho, \Pi]$* $\mathsf{implements}$ *the computation of a similarity function* SIM *if*

$$\forall q \in \mathcal{P}(E) \forall \hat{c} \in \hat{C} : \alpha_2(c) = \mathsf{SIM}(q, \hat{c}) \text{ for an initial activation based on } q$$

□

Note that in Definition 5.4 we have to distinguish between a case $\hat{c}$ and its node c within the BCRN. According to the discussion in Section 4.3, however, we can rely on an unequivocal mapping between the two and, thus, consider both as referring to the same object.

Composite Similarity Measures

In many applications, cases c are represented using attribute vectors

$$\hat{c} = [\hat{c}^1, \ldots, \hat{c}^k]$$

where each $\hat{c}^i$ represents a certain value of the ith attribute. In these situations, similarity of cases is often computed on the basis of the case components, e.g. by using a weighted sum of the *local* similarities to come up with a *global* similarity (Goos 1994; Wess 1995). As we shall see, BCRNs can model any such *composite* similarity measure:

Definition 5.5 (Composite similarity measure) *A similarity measure* SIM *is called* composite *if it is combined by a function* $\phi : \mathcal{R}^k \to \mathcal{R}$ *from feature similarity functions*

$$\mathsf{sim}^i : \mathsf{domain}_i \times \mathsf{domain}_i \to [0,1]$$

such that

$$\begin{aligned} \mathsf{SIM}(x,y) &= \mathsf{SIM}([x^1,\ldots,x^k],[y^1,\ldots,y^k]) \\ &= \phi(\mathsf{sim}^1(x^1,y^1),\ldots,\mathsf{sim}^k(x^k,y^k)) \end{aligned}$$

□

A simple example of a widely used composite similarity function is given by the weighted sum of the feature similarities:

$$\mathsf{SIM}([x^1,\ldots,x^k],[y^1,\ldots,y^k]) = \sum_{i=1,\ldots,k} w_i \cdot \mathsf{sim}^i(x^i,y^i). \tag{5.3}$$

If on the representation level SIM_{Rep} we are able to encode our similarity relationships by means of a composite function, a BCRN can be constructed that directly implements this model of similarity. For now, we will restrict ourselves to finite domains, i.e. we assume that we have a finite number of attributes each of which has again a finite domain. Then, the set E of IEs representing all these attribute values will also be finite. In addition, the case base $\hat{C}$ is assumed to be finite, too.

Theorem 5.2 *For any finite domain for every composite similarity function a BCRN can be constructed that implements that similarity function.*

Proof:
Let SIM be an arbitrary composite similarity measure with

$$\mathsf{SIM}(q,c) = \mathsf{SIM}([q^1,\ldots,q^k],[c^1,\ldots,c^k]) = \phi(\mathsf{sim}^1(q^1,c^1),\ldots,\mathsf{sim}^k(q^k,c^k))$$

Let further

- $E = \{e_1,\ldots,e_s\}$ be the set of all IEs occurring in the (finite) domain each representing a single attribute value
- $\hat{C}$ be the set of cases of the domain

Then we define a BCRN $N_{SIM} = [E, C, \sigma, \rho, \Pi]$ with

- C is the set of case nodes representing the cases in $\hat{C}$.
- For every pair $e_i, e_j \in E$ similarity will only be non-zero if both belong to the same attribute:

$$\sigma(e_i,e_j) = \begin{cases} \mathsf{sim}^l(e_i,e_j) & : \quad type(e_i) = A_l \wedge type(e_j) = A_l \\ 0 & : \quad \text{else} \end{cases}$$

- For every IE $e \in E$ and case node $c \in C$ the relevance will only be non-zero if e is contained in $\hat{c}$:

$$\rho(e,c) = \begin{cases} 1 & : \quad type(e) = A_l \wedge \hat{c} = [\hat{c}^1,\ldots,\hat{c}^k] \wedge e = \hat{c}^l \\ 0 & : \quad \text{else} \end{cases}$$

- For every IE $e \in E$ the propagation function computes the maximal input:

$$\pi_e(r_1,\ldots,r_s) = \mathsf{max}\{r_1,\ldots,r_s\}$$

- For every case node $c \in C$ the propagation function only takes into account the activations of those IEs that are contained in $\hat{c}$:

$$\pi_c(r_1,\ldots,r_s) = \phi(r_{i_1},\ldots,r_{i_k})$$

where $\rho(e_{i_j}, c) \neq 0$.

- For every IE $e \in E$ the initial activation will be based on q:

$$\alpha_0(e) = \begin{cases} 1 & : \quad e \in q \\ 0 & : \quad \text{else} \end{cases}$$

Given a BCRN defined in such a way as well as a query $q = [q^1, \ldots, q^k]$ and an arbitrary case $\hat{c} = [\hat{c}^1, \ldots, \hat{c}^k]$ which is represented in the BCRN by a case node c and IEs $e_{c^1}, \ldots, e_{c^k}$ $(q^i, e_{c^i} \in E)$, the activation of each e_{c^i} after similarity propagation will be as follows:

$$\begin{aligned}
\alpha_1(e_{c^i}) &= \pi_{e_{c^i}}(\sigma(e_1, e_{c^i}) \cdot \alpha_0(e_1), \ldots, \sigma(e_s, e_{c^i}) \cdot \alpha_0(e_s)) \\
&= \mathsf{max}\{\sigma(e_1, e_{c^i}) \cdot \alpha_0(e_1), \ldots, \sigma(e_s, e_{c^i}) \cdot \alpha_0(e_s)\} \\
&= \mathsf{max}\{\sigma(e_{q^1}, e_{c^i}) \cdot \alpha_0(e_{q^1}), \ldots, \sigma(e_{q^k}, e_{c^i}) \cdot \alpha_0(e_{q^k})\} \\
&= \sigma(e_{q^i}, e_{c^i}) \cdot \alpha_0(e_{q^i}) \\
&= \sigma(e_{q^i}, e_{c^i}) \\
&= sim^i(q^i, \hat{c}^i)
\end{aligned}$$

Given this, the activation of the case node c representing $\hat{c}$ after relevance propagation will be:

$$\begin{aligned}
\alpha_2(c) &= \pi_c(\rho(e_1, c) \cdot \alpha_1(e_1), \ldots, \rho(e_s, c) \cdot \alpha_1(e_s)) \\
&= \pi_c(0, \ldots, 0, \rho(e_{c^1}, c) \cdot \alpha_1(e_{c^1}), 0, \ldots, 0, \rho(e_{c^k}, c) \cdot \alpha_1(e_{c^k}), 0, \ldots) \\
&= \pi_c(0, \ldots, 0, \alpha_1(e_{c^1}), 0, \ldots, 0, \alpha_1(e_{c^k}), 0, \ldots) \\
&= \pi_c(0, \ldots, 0, \mathsf{sim}^1(q^1, \hat{c}^1), 0, \ldots, \mathsf{sim}^k(q^k, \hat{c}^k), 0, \ldots) \\
&= \phi(\mathsf{sim}^1(q^1, \hat{c}^1), \ldots, \mathsf{sim}^k(q^k, \hat{c}^k)) \\
&= \mathsf{SIM}([q^1, \ldots, q^k], [\hat{c}^1, \ldots, \hat{c}^k]) \\
&= \mathsf{SIM}(q, \hat{c})
\end{aligned}$$

Thus, the activation of the case node c equals the similarity of the case $\hat{c}$ as defined by SIM and, hence, the BCRN constructed this way implements the computation of SIM. ◇

Theorem 5.2 not only shows that any composite similarity measure can be represented in a BCRN but also that the retrieval process is complete and correct in the sense as discussed by Wess (1995, p. 161): Any case node in the net will have an activation α_2 according to the similarity of the case to the query which caused the initial activation. As, according to Theorem 5.1, α_2 is used to establish the preference orderings, completeness and correctness can be guaranteed.

General Similarity Measures

Of course, the limitation to composite similarity measures is a very strong one. However, BCRNs can, in principle, implement *any* similarity measure.

Theorem 5.3 *For any finite domain, every similarity function*

$$\mathsf{SIM}(q, \hat{c}) = \mathsf{SIM}(\{q^1, \ldots, q^l\}, \{\hat{c}^1, \ldots, \hat{c}^k\})$$

can be implemented by a BCRN.

Proof:
Let

- SIM be an arbitrary similarity function as described above
- $E = \{e_1, \ldots, e_s\}$ be the set of all IEs occurring in the (finite) domain
- $\hat{C}$ be the set of cases of the domain

Then we define a change of representation for cases and queries in that we do no longer consider each of them to be a set of IEs but rather a boolean vector of size s. This change of representation is achieved for a case $\hat{c}$ represented by a case node c and IEs $c^1, \ldots, c^k$ (in short: $c = \{c^1, \ldots, c^k\}$) as follows:

$$\dot{c} = [\dot{c}^1, \ldots, \dot{c}^s] \text{ where } \dot{c}^i = \begin{cases} 1 & : \quad c^i \in c \\ 0 & : \quad \text{else} \end{cases}$$

and similarly for any query $q = \{q^1, \ldots, q^l\}$. Based on this new representation, we can define $\dot{\mathsf{SIM}}$ as follows:

$$\begin{aligned} \dot{\mathsf{SIM}}(\dot{q}, \dot{c}) &= \dot{\mathsf{SIM}}([\dot{q}^1, \ldots, \dot{q}^s], [\dot{c}^1, \ldots, \dot{c}^s]) \\ &= \mathsf{SIM}(\{q^1, \ldots, q^l\}, \{\hat{c}^1, \ldots, \hat{c}^k\}) \\ &= \mathsf{SIM}(q, \hat{c}) \end{aligned}$$

Given this, we define a BCRN $N_{SIM} = [E, C, \sigma, \rho, \Pi]$ with

- C is the set of case nodes representing the cases in $\hat{C}$.
- $\forall e_i, e_j \in E$:
$$\sigma(e_i, e_j) = \begin{cases} 1 & : \quad i = j \\ 0 & : \quad else \end{cases}$$
- $\forall e \in E : \forall c \in C : \rho(e, c) = 1$
- $\forall e \in E : \pi_e(r_1, \ldots, r_s) = \mathsf{max}\{r_1, \ldots, r_s\}$
- $\forall c \in C : \pi_c(r_1, \ldots, r_s) = \dot{\mathsf{SIM}}(\{r_1, \ldots, r_s\}, \dot{c})$
- For every IE $e \in E$:
$$\alpha_0(e) = \begin{cases} 1 & : \quad e \in q \\ 0 & : \quad \text{else} \end{cases}$$

Then, we have for an arbitrary case node c

$$\begin{aligned} \alpha_2(c) &= \pi_c(\rho(e_1, c) \cdot \alpha_1(e_1), \ldots, \rho(e_s, c) \cdot \alpha_1(e_s)) \\ &= \pi_c(\alpha_1(e_1), \ldots, \alpha_1(e_s)) \\ &= \pi_c(\alpha_0(e_1), \ldots, \alpha_0(e_s)) \\ &= \dot{\mathsf{SIM}}(\{\alpha_0(e_1), \ldots, \alpha_0(e_s)\}, \dot{c}) \\ &= \dot{\mathsf{SIM}}(\dot{q}, \dot{c}) \\ &= \mathsf{SIM}(q, \hat{c}) \end{aligned}$$

◇

Basically, what happens is that all IEs representing the query are directly passed on to each case node, and the similarity of each case to the query is then computed independently of the other case nodes. So, in fact, this would be a somewhat awkward implementation of a linear search through case memory.

The situation was different for composite similarity measures as shown in the proof of Theorem 5.2. Here, the nature of a distributed representation has been utilized for example by computing the similarity of any pair of IEs just once and by propagating the resulting value to all associated case nodes.

Consequently, the question is not which similarity measures can be implemented by means of a BCRN but rather which can be implemented *efficiently*. We will address issues of efficiency in the following in more detail.

5.3.2 Efficiency

As mentioned before, Case-Based Reasoning systems have to be able to handle large case bases efficiently if they are to be used successfully in real world applications. Hence, the question arises as to what the effort for case retrieval is in a BCRN.

Considering Computational Complexity

A theoretical evaluation of the retrieval effort appears to be difficult as this depends on a number of parameters. As we have seen in the previous section already, the choice of a particular similarity measure will heavily influence the performance as this decision predetermines to what degree the distributed representation will be a benefit.

But even if a composite similarity measure is used, a number of factors influence the effort required for the retrieval process, including:

Size of the query: The more IEs have to be activated initially, the more has to be propagated through the net.

Degree of connectivity of IEs: The more non-zero similarity arcs exist, the more effort is required during similarity propagation.

Specificity of the IEs: The more cases are associated to the IEs, the more effort is required during relevance propagation.

Distribution of cases: If a large number of similar cases exists, many of these will become highly activated and thus retrieval effort will increase. Similarly, if only few similar cases exist, the scope of considered cases has to be extended until a sufficient number of cases has been found. Hence, a kind of *homogeneous distribution* might be desirable.

Desired number of cases: Recall that CRNs do not just retrieve cases but also provide a preference ordering. Hence, the more cases have to be retrieved, the more effort will be required even for sorting the retrieved cases.

Hence, we need to make some more assumptions in order to come up with at least a rough estimation of the computational effort. These assumptions are:

- Let SIM be a composite similarity measure based on k attributes in the domain, i.e. $E = E_{A_1} \cup \ldots \cup E_{A_k}$.
- Let the query consist of k IEs, too.
- Let there be a *similarity fan out* of out_{EE}, that is assume that each $e \in E$ has non-zero similarity arcs to out_{EE} IEs, including e itself.
- Let there be a *relevance fan out* of out_{EC}, that is assume that each $e \in C$ has non-zero relevance arcs to out_{EC} case nodes.

Given this, each step of the propagation process as described in the schema on page 44 requires the following effort:

Step 1 – Initial Activation:
each element of the query is mapped to an IE in the net:

$$k$$

Step 2 – Similarity Propagation:
for each initially activated IE, propagation is performed along the similarity arcs:

$$k \cdot out_{EE}$$

Step 3 – Relevance Propagation:
for each of the IEs activated in the previous steps, propagation is performed along the relevance arcs:

$$(k \cdot out_{EE}) \cdot out_{EC}$$

Furthermore, if there are $|E|$ IEs, $|C|$ cases and k attributes, then there will be on average $\frac{|E|}{k}$ IEs per attribute and as each case has exactly k values, i.e. k IEs associated to it, the relevance fan out will be

$$out_{EC} = \frac{|C| \cdot k}{|E|}$$

Putting it all together, the average computational effort for the entire retrieval process is:

$$\begin{aligned} O(\mathsf{Retrieve}) &= k + k \cdot out_{EE} + (k \cdot out_{EE}) \cdot out_{EC} \\ &= k + k \cdot out_{EE} + (k \cdot out_{EE}) \cdot \frac{|C| \cdot k}{|E|} \\ &= k + k \cdot out_{EE} + k^2 \cdot \frac{out_{EE}}{|E|} \cdot |C| \\ &= O(\frac{out_{EE}}{|E|} \cdot |C|) \end{aligned}$$

Consequently, retrieval in a CRN shows, in principle, a complexity linear in the size of the case base. However, the factor $\frac{out_{EE}}{|E|}$ will be very small in most applications. Also, adding new cases but leaving E unchanged is only possible to a certain extent without adding identical cases more than once. Hence, when enlarging the set of cases C significantly (for example when considering a larger product spectrum in an electronic commerce application), then one usually also adds new IEs and, thus, the effort will be less than linear.

Furthermore, the result shows that the similarity fan out out_{EE} is of crucial importance for efficiency. In fact, a *worst case* situation occurs if each IE representing a certain feature value has non-zero similarity arcs to all other IEs representing other values of the same feature.

Experimental Performance Results

Estimating the computational effort as described above requires a number of assumptions that appear to be unrealistic for many domains. Also, the value for the similarity fan out out_{EE} will depend on the targeted application, such that a general statement is impossible. To overcome these problems, a number of experiments have been performed to empirically test the efficiency of CRNs.

The major challenge when planning these experiments was to access sufficiently large case bases. In many case-based applications only relatively few cases have been used: In a comparative study of several retrieval methods (Auriol, Althoff, Wess, and Dittrich 1994), the largest case base did consist of the 1,500 cases used in the CABATA prototype (Lenz 1993; Lenz 1994a). In another case study, Goos (1994) used up to 3,000 cases.

In order to have much larger case bases accessible, we performed several experiments with data from the *UCI Machine Learning Repository* (Murphy and Aha 1992). Here, data sets with up to 65,000 cases have been encoded as cases and used for experiments. The results have been very promising, in that performance was similar to that of a commercial database system up to a certain number of cases when memory swapping occurred. More details can be found in Lenz (1996a).

However, these tests did have a major drawback: The data sets used were not really case bases but more or less randomly generated data records. In particular, fairly meaningless similarity measures were applied just to test retrieval efficiency. Whether the retrieved cases were actually similar did not matter.

As mentioned above, the chosen similarity measure influences the structure of the CRN and thus the efficiency of retrieval. In real-world domains, for example, often a *clustering* of the data naturally arises. This will result in groups of IEs connected with each other with similarity arcs but having no similarity relationships to other groups of IEs. As should be clear from the

theoretical considerations, this will improve the retrieval behavior of CRNs. In the domains of the *UCI Repository* this did not happen due to the artificially constructed similarity measure.

Recently, the VIRTUAL TRAVEL AGENCY provided a better means for testing the efficiency of CRNs. This project is explained in detail in Chapter 8. What is important here is that during peak season up to 250,000 offers are stored in that case base and retrieval can be performed in about 1.3 seconds on a standard workstation (SPARCStation-20).

This is even more remarkable as the structure of the data implies what has been described above as the *worst case* for retrieval complexity, namely that, according to the defined similarity measure, attributes exist for which every pair of IEs has non-zero similarity (in the VIRTUAL TRAVEL AGENCY this is the `Date` feature). As a consequence, every case will have a non-zero similarity to a given query.

5.3.3 Flexibility

As mentioned in Section 1.1.3, not only efficiency should be a criterion when investigating a memory model for case retrieval. We think that flexibility is of similar importance. In particular, we will in the following consider flexibility in terms of:

a) the retrieval process itself;

b) the required properties of similarity functions for which retrieval can be implemented by means of a CRN;

c) the maintenance effort required, for example for inserting new cases or modifying the similarity model;

d) opportunities to extend the model to cope with specific applications in a better way.

a) Flexible Case Retrieval

Most importantly, Case Retrieval Nets can be used to implement the process of information completion as described in Chapter 3. There is no need to distinguish between a problem and a solution part of a case. Rather, a case is simply represented as a set of all the information entities that belong to that case. Of course, it is possible to explicitly label a particular element as a solution, for example when performing classical case-based classification as defined in Section 2.2.6.

Compared in particular to *top-down* retrieval methods, such as kd–trees, there is no predefined ordering in which the symptoms in a diagnostic application, for example, have to be entered. Hence: *the memory structure itself does not limit in any way the retrieval process.*

Last but not least, missing information does not increase the complexity of retrieval: Similarity computations in BCRNs are *pessimistic* as long as unspecified IEs are not activated by the query activation. In this sense, missing values are treated as *don't know* values rather than *don't care*.

b) Properties of Similarity Functions

As shown in Theorem 5.3, basically any similarity function can be implemented by a Case Retrieval Net. However, it seems particularly suitable for composite functions where the distributed representation appears to be most beneficial.

The (global) similarity functions do not have to be symmetrical; even the (local) feature similarities are not necessarily symmetrical, if they are then two similarity arcs are required, one for each direction. Variations of the query activation α_0 permit a finer tuning of queries, e.g. in terms of dynamic weighting (see Section 5.5.1 below).

In many applications, a (global) composite similarity function is not sufficient. For example, the weights in a weighted sum may have to be adapted to special *contexts*. Usually, each different similarity function has to be computed separately. CRNs, on the contrary, provide mechanisms even for computing case-dependent similarity measures. By specifying appropriate propagation functions π_c for each case node $c \in C$, every such case node may even evaluate its input differently according to Equation 5.2 and thus assign other preferences and weights than different case nodes.

c) Maintenance Effort

BCRNs have a simple modular structure which is easy to understand, to implement, and to modify. Extensions are possible at any time, especially the number of features/attributes is not fixed — new attributes may be introduced simply by inserting related additional IE nodes. Also, modifications to the similarity relationships may be implemented by simply re-labeling the similarity arcs between the affected IEs.

Similarly, new cases are added by inserting new case nodes, additional IE nodes (as far as necessary) and specific labeled arcs. The complexity of this process depends mainly on the complexity of the new case – no rebuilding of the entire structure is required. Hence, BCRNs can be updated incrementally without losing the benefits of the structure.

d) Extensions of BCRNs

As with other net methods, the Case Retrieval Net model has a great potential for parallel work. In fact, the *Virtual Travel Agency* described in detail in Chapter 8 already makes use of several Case Retrieval Nets. There is no reason why these should not be managed in a distributed manner and run on different machines should issues of efficiency or the amount of data require this.

Finally, BCRNs offer the possibilities of extensions in various ways, some of these will be sketched in Chapter 6.

5.3.4 Similarity as Acceptance

While a general assumption is that the choice of a retrieval technique only has consequences with respect to efficiency, we claim here that there are principle differences which should be taken into account during system design.

As discussed in detail by Burkhard (1998a) a crucial difference can be observed with respect to the differences of a *top-down* and a *bottom-up* retrieval process. This difference is not so much of a cognitive nature but has consequences for the characteristics of representable similarity measures.

In the following we will illustrate these differences for systems using a feature-value representation of cases and implementing a composite similarity measure: Given a set of information entities E and having cases c and queries q represented as[1]

$$c = [c^1, \ldots, c^k] \text{ and } q = [q^1, \ldots, q^k] \ (c^i, q^i \in E_i, E = \cup E_i)$$

the (global) similarity $\mathsf{SIM}(q, c)$ is assumed to be computed by means of some function $\mathcal{F}$ based on the *local* similarities $\mathsf{sim}^i(q^i, c^i)$:

$$\mathsf{SIM}(q, c) = \mathcal{F}(\mathsf{sim}^1(q^1, c^1), \ldots, \mathsf{sim}^k(q^k, c^k)) \tag{5.4}$$

[1] To improve readability, we will from now on no longer distinguish between cases $\hat{c_i}$ and their corresponding case nodes c_i but assume that both refer to the same object as discussed before.

Retrieval by Rejection

As mentioned above, all cases are initially considered to be potentially relevant in a *top-down* retrieval process. By checking certain conditions (such as testing the values of certain attributes in a decision tree like model), the set of potentially relevant cases is subsequently reduced. Hence, this type of retrieval can be referred to as *retrieval by rejection*.

Rejection of cases is, in general, based on isolated properties, such as values of single features, that is *local similarities*, are used to discard cases. Of course, one could in principle also consider multiple features but due to reasons of efficiency most often just a single feature is used. In the extreme case, one could even compute the (global) similarity, then the memory structure would become obsolete and, in fact, a simple linear search would be performed right in the root of the top-down structures.

When the entire retrieval process is completed, cases remain which appear to be sufficiently similar to the given problem description according to every single feature. A single feature in which the case and the problem differ significantly is sufficient for eliminating this case from the set of potentially relevant cases. Thus, similarity of top-down retrieval corresponds to the intuitive understanding of a *distance function*: A sufficiently large distance in one dimension will cause a large overall distance. To summarize:

Observation 5.1 *Top-down retrieval procedures retrieve cases which are similar to the query with respect to* all *of the given features.*

When using a composite distance function $DIST$, Equation 5.4 would read as:

$$DIST(q,c) = \mathcal{F}(dist^1(q^1,c^1),\ldots,dist^k(q^k,c^k)) \tag{5.5}$$

Retrieval by Accumulation

The situation is different when performing a retrieval process in a bottom-up fashion. Here, the process starts with a kind of *seed* which corresponds to the problem description, or query. By subsequently *extending* the scope in all dimensions, that is by taking into account less similar values for all features, cases that have been described by means of these feature values are *included* in the set of potential relevant cases. This type of retrieval can be referred to as *retrieval by accumulation*.

Here, a single feature in which the case and the query are similar to each other suffices for the inclusion of the case. Consequently, a more *holistic* view on similarity is implemented which describes the *acceptability* of cases.

Observation 5.2 *Bottom-up retrieval procedures can retrieve cases which are similar to the query with respect to* at least one *of the given features.*

A composite accumulation function ACC can be implemented on the basis of the local ones as usual:

$$ACC(q,c) = \mathcal{F}(acc^1(q^1,c^1),\ldots,acc^k(q^k,c^k)) \tag{5.6}$$

Rejection and Accumulation: A Comparison

Example 5.1 To understand the difference with respect to distance and accumulation functions consider an example where cases and queries have just two feature values and the similarity is computed by an accumulation function as:

$$ACC(q,c) = ACC([q^1,q^2],[c^1,c^2]) = \frac{acc^1(q^1,c^1) + acc^2(q^2,c^2)}{2}$$

that is the average of the local similarities will be computed.
If a single feature of the case is now absolutely dissimilar to the query's value, say $\mathsf{sim}^1(q^1,c^1) = 0$, the overall value for the accumulation function $ACC(q,c)$ can still become as high as 0.5 if $acc^2(q^2,c^2) = 1$. Thus, despite dissimilarities in single features, a certain degree of similarity can still be achieved.
When translating this example to distance functions and retrieval by rejection, the situation changes: As distances and similarities are considered to be inverse measures, the local distances for this example would become:

$$dist^1(q^1,c^1) = \mathsf{max} \text{ and } dist^2(q^2,c^2) = 0$$

where max is some maximal value for the (local) distance. Given this, an intuitive assumption would be that for the global distance $DIST(q,c) = \mathsf{max}$ holds which expresses a total dissimilarity of q and c.

Admittedly, this example is a bit misleading in so far as it assumes more than is defined, for example, for a distance function in mathematical theory. It may well be possible that one can define a mapping between similarity and distance functions that does not show the above behavior. This, however, would then probably violate some intuitive assumptions about distances, such as that the global distance cannot be lower than the largest local one.

At any rate, the above examples shows that there is a difference at least in situations where cases are represented as feature vectors and a weighted sum is used as a (composite) similarity measure both for similarities as well as for distances.

The problem becomes even more obvious if we display the characteristic functions resulting from typical accumulation and distance functions graphically. If one plots the lines representing those cases which have equal *distance* to a query (q_x, q_y), where distance is computed as the Manhattan distance

$$DIST([q^1,q^2],[c^1,c^2]) = |c^1 - q^1| + |c^2 - q^2|$$

one obtains the picture displayed in Figure 5.2.

If we translate the Manhattan distance into local accumulation functions, for example by using one of the formulas suggested by Wess (1995, p. 53) for the translation of distance and similarity functions:

$$acc(x,y) := \frac{1}{1 + dist(x,y)} \tag{5.7}$$

we obtain

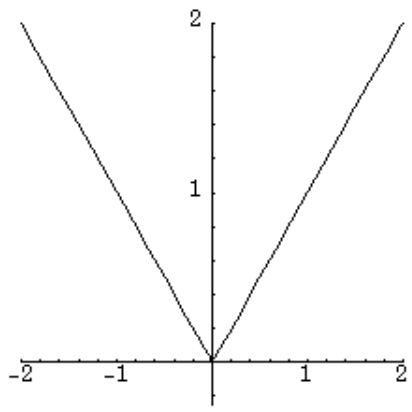
$$acc_x(c_x,q_x) = \frac{1}{1+|c_x - q_x|} \text{ and } acc_y(c_y,q_y) = \frac{1}{1+|c_y - q_y|}$$

and could compute

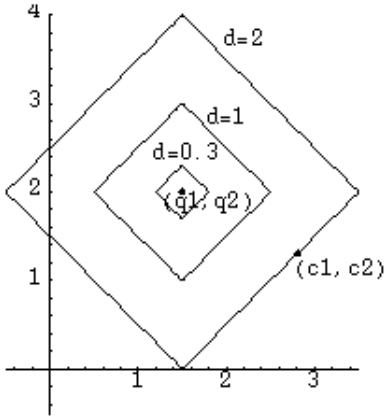
$$ACC(c,q) = \frac{acc_x(c_x,q_x) + acc_y(c_y,q_y)}{2}$$

which would result in lines of cases having equal accumulation levels as plotted in Figure 5.3.

As Figures 5.2 and 5.3 indicate, both approaches result in different plots. It is crucial that this difference is not due to the transformation applied according to Equation 5.7. Instead, the

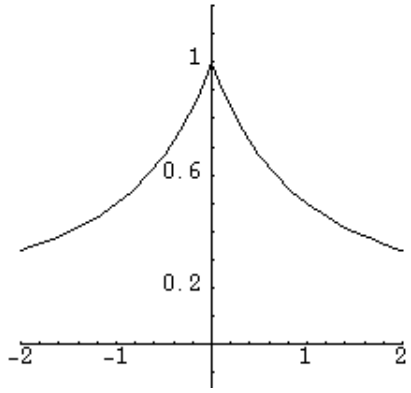


$f(x) = |x|$

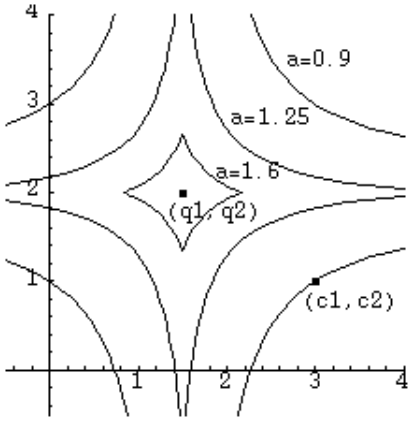


$|q^1 - c^1| + |q^2 - c^2| = d$

Figure 5.2: Characteristics of composite distance (adapted from Burkhard (1998)).



$f(x) = 1/(1 + |x|)$



$1/(1 + |q_1 - c_1|) + 1/(1 + |q_2 - c_2|) = a$

Figure 5.3: Characteristics of composite accumulation functions (adapted from Burkhard (1998)).

difference is due to the definition of similarity by accumulation: Despite single feature values being dissimilar, a case can still be similar to a query with respect to the global similarity measure. Hence the characteristics of accumulation functions do not necessarily form bounded areas. Burkhard (1998a) referred to this as the ability of making compromises.

To summarize, *top-down* retrieval is performed by rejecting cases whereas *bottom-up* retrieval collects cases by accumulation. As shown above, there are significant differences with respect to the retrieval results.

In the remainder of this thesis, we will return to the more common notion of *similarity* but when not expressed otherwise we assume similarity computed in the sense of retrieval by accumulation rather than by rejection.

5.4 Limitations

5.4.1 Retrieval Based on Adaptability

BCRNs do not support directly the integration of adaptation knowledge in the retrieval phase. As the name indicates, this memory model may be used to *retrieve* cases – what these case are used for is up to another system or the user. However, as for all retrieval methods, this may in some situations result in cases being retrieved which appear to be sub-optimal in the sense that the most *similar* cases are not necessarily the ones that are *easiest adaptable* to the current problem (cf. Smyth and Keane 1993).

5.4.2 Structural Similarities

CRNs have been designed for finding cases with components similar to a query. In particular, CRNs inherit some ideas of a *distributed representation* as introduced by the *Parallel Distributed Processing* group (Rumelhart, McClelland, and the PDP Research Group 1986). This means that an IE node encoding a certain property of cases is present just once – no matter how many cases share this property.

Consequently, the encoding of cases within a CRN is crucial. More precisely, the representation of cases is critical because similarity will be assessed based on these representations. Hence, when designing a CBR system making use of CRNs, one should have in mind a certain *scenario*, i.e., what types of queries will the potential user of such a system enter, what might s/he consider as similar, etc. — or, in terms of Burkhard (1998a): What would the user *accept* as suitable cases?

For some applications, this may partially be avoided by extending the net with additional structures, such as in *Object-directed Case Retrieval Nets* (cf. Section 6.3). Here only a part of the entire inference process is performed inside the CRN while other parts are carried out in more structured memory models.

5.5 Minor Extensions of BCRNs

After having described in detail the model of Basic Case Retrieval Nets in this chapter, we will now sketch some extensions that do not really form new models (as is the case with the extensions described in Chapter 6) but are minor variations of BCRNs.

5.5.1 Dynamic Weighting

According to Definition 3.2, a query is just a set of IEs — that is, merely a collection of knowledge items that appear to be relevant in the current problem situation. In many applications, however, it is desirable to have means for dynamically weighting the various parts of the query, for example in order to express certain preferences or priorities. For example, the weights w_i in Equation (5.3) are often specified dynamically, i.e. they are given as part of the query thus expressing the current user's preferences.

To handle such situations we need to extend Definition 3.2 in order to allow for weighted queries:

Definition 5.6 (Weighted Query) *A* weighted query *is a set of pairs* $(e_i, w(e_i))$ *where* $e_i \in E$ *are IEs and* $w(e_i) \in [0,1]$ *are weighting factors expressing preferences of the given IEs.*

$$q = \{(e_{i_1}, w(e_{i_1})), \ldots, (e_{i_l}, w(e_{i_l}))\}$$

□

As Theorem 5.2 showed, *any* composite similarity measure can be represented in a BCRN. The problem with dynamic queries, however, is that, in fact, the similarity measure may be different for each query due to changing weights.
The proof of that theorem very much relied on putting very much knowledge into the propagation functions of the nodes. In particular, any kind of weighting would have to be performed internally in the propagation functions of the case nodes. Hence, the BCRN model could be extended in such a manner that not only simple values expressing similarity or relevance, respectively, are propagated through the net, but that pairs consisting of these values and the user-given weight are passed along the arcs. Then, the propagation function of each case node not only would consider the amount of activation that reaches that node but also the weights as expressed in the query.

However, for most situations, such an extension is not really required. A BCRN that is, in fact, much simpler and more flexible in this respect can be constructed if the global similarity function ϕ is linear in every argument:

$$\forall l = 1, \ldots, k : \phi(r_1, \ldots, a \cdot r_l, \ldots, r_k) = a \cdot \phi(r_1, \ldots, r_l, \ldots, r_k)$$

We will show this in the following for a commonly used composite similarity function, namely a weighted sum:

Theorem 5.4 *For any finite domain, a BCRN can be constructed that implements the similarity function*

$$\mathsf{SIM(q,c)} = \mathsf{SIM}([q^1, \ldots, q^k], [c^1, \ldots, c^k]) = \sum_{i=1,\ldots,k} w(q^i) \cdot \mathsf{sim}^i(q^i, c^i)$$

where the weights $w(q^i)$ can be specified as part of the query.

Proof:
Let

- $E = \{e_1, \ldots, e_s\}$ be the set of all IEs occurring in the (finite) domain
- $\hat{C}$ be the set of cases of the domain

Then we define a BCRN $N = [E, C, \sigma, \rho, \Pi]$ as follows:

- C is the set of case nodes representing the cases in $\hat{C}$.
- For every pair $e_i, e_j \in E$ let (as in Theorem 5.2):

$$\sigma(e_i, e_j) = \begin{cases} \mathsf{sim}^l(e_i, e_j) & : \quad type(e_i) = A_l \wedge type(e_j) = A_l \\ 0 & : \quad \text{else} \end{cases}$$

- For every IE $e \in E$ and case node $c \in C$

$$\rho(e, c) = \begin{cases} 1 & : \quad type(e) = A_l \wedge \hat{c} = [\hat{c}^1, \ldots, \hat{c}^k] \wedge e = \hat{c}^l \\ 0 & : \quad \text{else} \end{cases}$$

- For every IE $e \in E$

$$\pi_e(r_1, \ldots, r_s) = \mathsf{max}(r_1, \ldots, r_s)$$

- For every case node c

$$\pi_c(r_1, \ldots, r_s) = \sum_{i=1,\ldots,s} r_i$$

- Set the initial activation according to weights expressed in the query:

$$\alpha_0(e) = \begin{cases} w(e) & : \quad (e, w(e)) \in q \\ 0 & : \quad \text{else} \end{cases}$$

Given this, we can determine what the activation of an arbitrary case node $c \in C$ representing a case $c = [c^1, \ldots, c^k]$ will be for a given query $q = [(q^1, w(q^1)), \ldots, (q^k, w(q^k))]$ (for simplicity we assume here that the query is a set of pairs each consisting of an IE and a weight of that IE):

$$\begin{aligned}
\alpha_2(c) &= \pi_c(\rho(e_1, c) \cdot \alpha_1(e_1), \ldots, \rho(e_1, c) \cdot \alpha_1(e_1)) \\
&= \sum_{i=1,\ldots,s} \rho(e_i, c) \cdot \alpha_1(e_i) \\
&= \sum_{i=1,\ldots,k} \rho(e_{c^i}, c) \cdot \alpha_1(e_{c^i}) \\
&= \sum_{i=1,\ldots,k} \alpha_1(e_{c^i}) \\
&= \sum_{i=1,\ldots,k} \pi_{e_{c^i}}(\sigma(e_1, e_{c^i}) \cdot \alpha_0(e_1), \ldots, \sigma(e_s, e_{c^i}) \cdot \alpha_0(e_s)) \\
&= \sum_{i=1,\ldots,k} \mathsf{max}\{\sigma(e_1, e_{c^i}) \cdot \alpha_0(e_1), \ldots, \sigma(e_s, e_{c^i}) \cdot \alpha_0(e_s)\} \\
&= \sum_{i=1,\ldots,k} \sigma(e_{q^i}, c^i) \cdot \alpha_0(e_{q^i}) \\
&= \sum_{i=1,\ldots,k} \mathsf{sim}^i(q^i, c^i) \cdot w(q^i) \\
&= \mathsf{SIM}(q, c)
\end{aligned}$$

◇

5.5.2 Computational Nodes

In all the discussions in this chapter, a crucial assumption was that the domain is finite. For most applications, this is an unrealistic assumption in so far as

- the number of cases stored at any time is, indeed, finite;
- the number of IEs associated to these cases is finite, too;
- but a query may contain a value that is not yet represented by an IE in the CRN.

As an example consider a feature like `Price` in an electronic commerce application: A user of such a system might enter an arbitrary positive value which may not be represented in the net. Hence, an extension of BCRNs is required which allows it to handle features with infinitely many values (and possibly also features with large ranges where it does not seem reasonable to explicitly represent all potential values as IEs in the net). We will call such feature an *infinite* feature in the following.

For this, a fairly simple extension of the BCRN model is sufficient which we will call *computational nodes* (Burkhard 1995a; Burkhard 1998b). The idea is to introduce another kind of node into the net structure which uses the defined local similarity function for a single infinite feature to determine the appropriate activation levels of all IE nodes in the net that represent values of the same feature (and are associated to cases — that's why they are present in the net).

The 3–Step–Scheme from page 44 would then be modified to:

Let $q = [q^1, \ldots, q^k], q^i \in \mathsf{domain}(A_i)$ be a query.

Step 1 – Initial Activation:
For each q_i:

- If A_i is an infinite feature, then assign q^i to the computational node of A_i
- else determine α_0 for the IE representing q^i.

Step 2 – Similarity Propagation:
For each $e \in E$, let $A_i = type(e)$:

- If A_i is an infinite feature, then

$$\alpha_1(e) = ComputationalNode_{A_i} :: ComputeSim(q^i, e)$$

- else

$$\alpha_1(e) = \pi_e(\sigma(e_1, e) \cdot \alpha_0(e_1), \ldots, \sigma(e_s, e) \cdot \alpha_0(e_s))$$

Step 3 – Relevance Propagation:
As usually, propagate to case nodes:

$$\alpha_2(c) = \pi_c(\rho(e_1, c) \cdot \alpha_1(e_1), \ldots, \rho(e_s, c) \cdot \alpha_1(e_s))$$

Figure 5.4 displays the general architecture of a BCRN which has been extended by computational nodes. Obviously, adding computational nodes does not change the principle properties of BCRNs. In particular,

- completeness and correctness can still be guaranteed in the sense of Definition 5.4.
- efficiency is not significantly changed since we realized in Section 5.3.2 that most of the work has to be done during the third step, relevance propagation. This, however, remains unchanged.

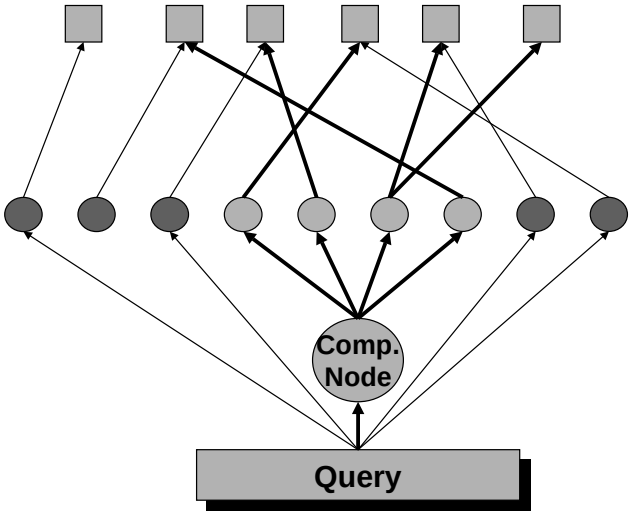


Figure 5.4: Architecture of a BCRN with computational nodes. The dark circles represent IEs of the usual manner while the lighter ones represent the values of an infinite feature that occur in the case base. A computational node is used to activate these IEs for a given query according to the specified (local) similarity function for that feature.

Typical examples where computational nodes are useful are situations in which numeric features have to be handled that have an infinite range, such as the `Price` example given above. However, in one of the many applications that CRNs have been used for, we have used computational nodes also for a symbolic feature where the local similarity for that feature had

been defined based on whether or not a regular expression given in the query would match a feature value or not. As with the numeric features above, the crucial property here is that the query space is infinite and, hence, cannot be represented in a pure BCRN.

Generally speaking, according to the schematic description above, computational nodes can be particularly useful if a composite similarity measure is used and, thus, the set of all IEs can be partitioned according to the features being used. However, this is not necessarily required. In an extreme situation, computational nodes might even be used if there is just a single set of IEs (that can not be partitioned) and the activation of *all* IEs after similarity propagation is determined by means of the computational node. Whether this would be a reasonable application of this model is a different issue.

Chapter 6

Extended Models of CRNs

One of the most frustrating and at the same time tantalizing things about working in AI is that you are never done.
Roger C. Schank, AI Magazine, 1984

After having introduced the basic model of CRNs, we will now present a number of possible extensions that appeared to be useful for particular tasks. These extensions address both a better structured memory as well as a retrieval procedure with anytime properties. Some of these extensions have been fully implemented while others are only ideas that require further evaluation.

6.1 Conceptual Case Retrieval Nets

So far, Basic Case Retrieval Nets have been described as a flat memory structure consisting of only two layers in the net: the IE nodes and the case nodes. In order to perform the retrieval process, we assumed in Section 5.2 that for each pair of IEs the similarity is explicitly specified by means of the similarity function σ. According to the graphical representation, this means that a labeled similarity arcs exists between every two IE nodes.

In certain circumstances, however, it might be desirable not to *explicitly* represent the relationships between the IEs but to use, for example, a conceptual taxonomy of IEs and, thus, to represent relationships implicitly.

In this section, an extension of the BCRN model will be presented which allows for modeling the similarity of IEs in such a way. Furthermore, it will be shown that, under certain circumstances, an equivalent BCRN can be constructed. Hence, the extended net can be used for easier modeling whereas BCRNs can be employed for the retrieval process itself.

6.1.1 The CCRN Model

Compared to the BCRN model, a Conceptual Case Retrieval Net is extended by integrating more abstract *concept nodes* K and a special *prototypicality function* δ describing the relationships between IEs and concepts:

Definition 6.1 (Conceptual Case Retrieval Net) *A* Conceptual Case Retrieval Net *(CCRN) is defined as a structure* $N = [E, C, K, \sigma, \rho, \delta, \Pi, t_*]$ *with*

E *is the finite set of* IE nodes*;*

C *is the finite set of* case nodes*;*

K *is the finite set of* concept nodes*;*

σ *is the* similarity function

$$\sigma : E \times E \to \mathcal{R}$$

which describes the similarity $\sigma(e_i, e_j)$ *between IEs* e_i, e_j*;*

ρ *is the* relevance function

$$\rho : E \times C \to \mathcal{R}$$

which describes the relevance $\rho(e, c)$ *of the IE* e *to the case node* c*;*

δ *is the* prototypicality function

$$\delta : E \cup K \times E \cup K \to \mathcal{R}$$

which describes the relationships between the nodes in the concept hierarchy, including the IE nodes;

Π *is the set of* propagation functions

$$\pi_n : \mathcal{R}^{E \cup K} \to \mathcal{R}, n \in E \cup K$$

and

$$\pi_c : \mathcal{R}^{E} \to \mathcal{R}, c \in E$$

describing how each node adjusts its internal state according to the activations of other nodes.

t_* *is the time when activation of case nodes is determined in order to establish a preference ordering of cases for a given query, i.e. this preference ordering is built according to*

$$\alpha_{t_*}(c), c \in C$$

□

Note that $[E, C, \sigma, \rho, \Pi]$ would be a BCRN with the exception that the propagation functions $\pi \in \Pi$ have an extended domain in a CCRN. Furthermore, Definition 6.1 requires some remarks:

- Firstly, one could define $E' = E \cup K$ and extend σ accordingly. Then, one would obtain the same structure as in a BCRN. However, the above distinction in a better way reflects the differences between IEs on the one hand (which are the atomic constituents of cases) and more abstract concepts on the other hand (which may be used to structure the space of all IEs). Hence, the $k_i \in K$ are assumed not to be present in the case representations.

- Similarly, the *prototypicality function* δ is used to specify relationships in terms of the conceptual hierarchy, such as inheritance or part-of relationships. As we will see later, these values are used in a similar way as are the values of σ during the retrieval process. Again, distinguishing these two functions provides better ways of modeling.

- Due to the assumption of concepts not being present in the cases, propagation functions of case nodes do not take into account the states of concept nodes but, as in BCRNs, depend only on the state of IE nodes.

- The parameter t_* is required because the spreading activation process through the entire net may require several time steps due to a complex concept node hierarchy, for example. Consequently, the activation of cases will change over time and, thus, it is essential to define which of these activations will be used for establishing the preference ordering that will be the result of the retrieval process.
- Finally, the above definition not necessarily requires that the concept nodes form a hierarchy. Rather, it is a finite directed graph. If, for example, a hierarchy is desired, than δ has to be specified appropriately.

6.1.2 Retrieval in CCRNs

Similarly to the propagation process in BCRNs, case retrieval in CCRNs is performed by propagating activations through the net.

Definition 6.2 (Activation of a CCRN) *An* activation *of a Conceptual Case Retrieval Net* $N = [E, C, K, \sigma, \rho, \delta, \Pi, t_*]$ *is a function*

$$\alpha : E \cup K \cup C \to \mathcal{R}$$

□

Exactly as for BCRNs (cf. Definition 5.2), these *activations* are annotations to the nodes. However, for computing these activations, the extended structure of the conceptual net has to be taken into account and, thus, three different types of propagation functions are in use:

Definition 6.3 (Propagation process in a CCRN) *Consider a Conceptual Case Retrieval Net* $N = [E, C, K, \sigma, \rho, \delta, \Pi, t_*]$ *with* $E = \{e_1, \ldots, e_s\}$ *and* $K = \{k_1, \ldots, k_l\}$. *Further, let*

$$\alpha_t : E \cup K \cup C \to \mathcal{R}$$

be the activation of the net at time t. *Given this, the activation at time* $t+1$ *is determined as follows:*

for IE nodes $e \in E$**:**

$$\begin{aligned} \alpha_{t+1}(e) \quad = \quad & \pi_e(\sigma(e_1, e) \cdot \alpha_t(e_1), \ldots, \sigma(e_s, e) \cdot \alpha_t(e_s), \\ & \delta(k_1, e) \cdot \alpha_t(k_1), \ldots, \delta(k_l, e) \cdot \alpha_t(k_l)) \end{aligned}$$

for concept nodes $k \in K$**:**

$$\begin{aligned} \alpha_{t+1}(k) \quad = \quad & \pi_k(\delta(e_1, k) \cdot \alpha_t(e_1), \ldots, \delta(e_s, k) \cdot \alpha_t(e_s), \\ & \delta(k_1, k) \cdot \alpha_t(k_1), \ldots, \delta(k_l, k) \cdot \alpha_t(k_l)) \end{aligned}$$

for case nodes $c \in C$**:**

$$\alpha_{t+1}(c) \quad = \quad \pi_c(\rho(e_1, c) \cdot \alpha_t(e_1), \ldots, \rho(e_s, c) \cdot \alpha_t(e_s))$$

The initial activation is determined based on the given query $q = \{q^1, \ldots, q^n\}$*:*

$$\alpha_0(n) = \begin{cases} 1 & : \quad n \in E \wedge n \in q \\ 1 & : \quad n \in K \wedge n \in q \\ 0 & : \quad \textit{otherwise} \end{cases}$$

□

Definition 6.3 is highly similar to Definition 5.3 which described the propagation process in a BCRN. There are only two differences: Firstly, as mentioned above, the propagation functions have to be extended to reflect the existence of concept nodes. Secondly, a query may not only contain IEs but also more abstract concepts. Cases, however, should also be represented as sets of IEs. Given these formal descriptions, the entire retrieval process in a CCRN can be described as follows:

Theorem 6.1 *Consider a CCRN* $N = [E, C, K, \sigma, \rho, \delta, \Pi, t_*]$ *with the activation function* α *as defined in Definitions 6.2 and 6.3. The result of the case retrieval for a given query is the* preference ordering *of cases according to decreasing activations* $\alpha_{t_*}(c)$ *of case nodes* $c \in C$.

6.1.3 An Illustration

Figure 6.1 shows an example of a Conceptual Case Retrieval Net for the VIRTUAL TRAVEL AGENCY domain: The IEs for the `Destination` feature are arranged according to geographic locations thus allowing for a similarity assessment based on common conceptual nodes (arcs representing zero similarity respectively prototypicality are omitted).

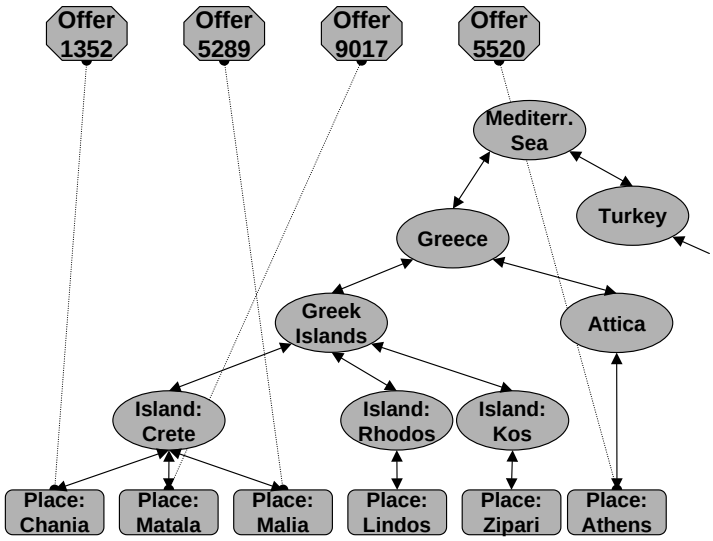


Figure 6.1: Example of a Conceptual CRN for the VIRTUAL TRAVEL AGENCY domain

6.1.4 Translation to BCRNs

While Conceptual CRNs provide a more structured case memory than Basic CRNs, the propagation process is more complicated because of two reasons: Firstly, three different types of nodes have to be considered. Secondly, the propagation process usually has to be performed over t_* time steps and cannot be finished after just 2 steps as in BCRNs.

Thus, it is an interesting question whether a CCRN can be *compiled* to a BCRN. More precisely, we will now investigate whether for any given CCRN N a BCRN N' can be constructed which is equivalent in the sense that, for any given query, both nets deliver the same preference orderings of cases. For the equivalence of the preference orderings it is obviously sufficient if the levels of activation of the case nodes after termination of the propagation process are identical:

Definition 6.4 (Equivalence of CRNs) *A BCRN $N' = [E', C', \sigma', \rho', \Pi']$ is* equivalent *to a Conceptual Case Retrieval Net $N = [E, C, K, \sigma, \rho, \delta, \Pi, t_*]$ iff $C = C'$ and*

$$\forall q \in \mathcal{P}(E) \forall c \in C : \alpha'_2(c) = \alpha_{t_*}(c)$$

□

But before we address the *compilation* of a CCRN to a BCRN, let us make clear that, even in a highly structured CCRN, the activation of IE nodes at each stage of the propagation process can be described by a function depending only on the initial activations caused by the query:

Lemma 6.1 *At any time during the propagation process, the activation of all IE nodes $e \in E$ and concept nodes $k \in K$ in a CCRN can be described as a direct function of the initial activation of IEs.*

Proof:
Let $N = [E, C, K, \sigma, \rho, \delta, \Pi, t_*]$ be a CCRN with $E = \{e_1, \ldots, e_s\}$ and $K = \{k_1, \ldots, k_l\}$. Let further be $e \in E$ an arbitrary IE node. According to Definition 6.3, the activation of e at time $t > 0$ is computed by

$$\alpha_t(e) \quad = \quad \pi_e(\sigma(e_1, e) \cdot \alpha_{t-1}(e_1), \ldots, \sigma(e_s, e) \cdot \alpha_{t-1}(e_s), \delta(k_1, e) \cdot \alpha_{t-1}(k_1), \ldots, \delta(k_l, e) \cdot \alpha_{t-1}(k_l))$$

We will prove the lemma now by induction over t:
$\underline{t = 1:}$
In the case of $t = 1$ we have the trivial situation that there exists a function

$$\mathcal{F}_e^1 : \mathcal{R}^{E \cup K} \to \mathcal{R}$$

such that

$$\begin{aligned} \alpha_t(e) &= \pi_e(\sigma(e_1, e) \cdot \alpha_0(e_1), \ldots, \sigma(e_s, e) \cdot \alpha_0(e_s), \delta(k_1, e) \cdot \alpha_0(k_1), \ldots, \delta(k_l, e) \cdot \alpha_0(k_l)) \\ &= \mathcal{F}_e^1(\alpha_0(e_1), \ldots, \alpha_0(e_s), \alpha_0(k_1), \ldots, \alpha_0(k_l)) \end{aligned}$$

In the following, we will use the expression $\alpha_0(E \cup K)$ as a short hand meaning

$$\alpha_0(E \cup K) = \alpha_0(e_1), \ldots, \alpha_0(e_s), \alpha_0(k_1), \ldots, \alpha_0(k_l)$$

$\underline{t \to t+1:}$
Assuming that for an arbitrary $t > 0$ there exists some function $\mathcal{F}_e^t$ with

$$\alpha_t(e) = \mathcal{F}_e^t(\alpha_0(E \cup K))$$

we can then show that there will also exist a function

$$\mathcal{F}_e^{t+1} : \mathcal{R}^{E \cup K} \to \mathcal{R}$$

such that

$$\begin{aligned} \alpha_{t+1}(e) &= \pi_e(\sigma(e_1, e) \cdot \alpha_t(e_1), \ldots, \sigma(e_s, e) \cdot \alpha_t(e_s), \\ &\qquad \delta(k_1, e) \cdot \alpha_t(k_1), \ldots, \delta(k_l, e) \cdot \alpha_t(k_l)) \\ &= \pi_e(\sigma(e_1, e) \cdot \mathcal{F}_{e_1}^t(\alpha_0(E \cup K)), \ldots, \sigma(e_s, e) \cdot \mathcal{F}_{e_s}^t(\alpha_0(E \cup K)), \\ &\qquad \delta(k_1, e) \cdot \mathcal{F}_{k_1}^t(\alpha_0(E \cup K)), \ldots, \delta(k_l, e) \cdot \mathcal{F}_{k_l}^t(\alpha_0(E \cup K))) \\ &= \mathcal{F}_e^{t+1}(\alpha_0(e_1), \ldots, \alpha_0(e_s), \alpha_0(k_1), \ldots, \alpha_0(k_l)) \\ &= \mathcal{F}_e^{t+1}(\alpha_0(E \cup K)) \end{aligned}$$

Thus, it is possible to describe the activation of an IE node e at a given time t by some function $\mathcal{F}_e^t$ which only depends on the initial activations. The existence of $\mathcal{F}_k^t$, $k \in K$ can be shown in an analogous way. ◇

Given Lemma 6.1, we can now return to the original objective of *compiling* a Conceptual CRN to a Basic CRN.

Theorem 6.2 *For any given CCRN $N = [E, C, K, \sigma, \rho, \delta, \Pi, t_*]$ an equivalent BCRN can be constructed.*

Proof:
We will prove the theorem by showing how the BCRN can be constructed for the given CCRN N. For this, let $N' = [E', C', \sigma', \rho', \Pi']$ be a BCRN with

- $E' = E \cup K$, $E' = \{e_1, \ldots, e_s, k_1, \ldots, k_l\}$
- $C' = C$
- $\sigma' : E' \times E' \to \mathcal{R}$ with
$$\forall n_i, n_j \in E' \sigma'(n_i, n_j) = 1$$
- $\rho' : E' \times C' \to \mathcal{R}$ with
$$\forall n \in E' \forall c \in C' \rho'(n, c) = \begin{cases} \rho(n, c) & : \quad n \in E \\ 0 & : \quad n \notin E \end{cases}$$
- $\forall n \in E' \pi'_n : \mathcal{R}^{E'} \to \mathcal{R}$
- $\forall c \in C' \pi'_c : \mathcal{R}^{E'} \to \mathcal{R}$ such that the propagation functions of case nodes are independent of the activation of formerly concept nodes:
$$\pi'_c(r_{e_1}, \ldots, r_{e_s}, r_{k_1}, \ldots, r_{k_l}) = \pi_c(r_{e_1}, \ldots, r_{e_s})$$

Given in particular the latter condition, it obviously, suffices to show that at time $t = 1$ the activations of all $e \in E$ in N' is the same as the activations of IE nodes at time $t = t_* - 1$ in N — given this and π'_c as defined above, the activation of case nodes will be identical, too.

Hence we have to specify $\pi'_n, n \in E'$ such that

$$\forall n \in E : \alpha_{t_*-1}(e) = \alpha'_1(e)$$

where α' denotes the activation of the nodes in N'. For this we can use Lemma 6.1 which showed that for all IE nodes a function $\mathcal{F}_e^{t_*-1}$ can be constructed such that

$$\forall n \in E : \alpha_{t_*-1}(e) = \mathcal{F}_e^{t_*-1}(\alpha_0(E \cup K))$$

If we set the initial activation of all nodes in N' as in N

$$\forall n \in (E' \cup C') : \alpha'_0(n) = \alpha_0(n)$$

and further define for all nodes $n \in E'$

$$\pi'_n(r_{e_1}, \ldots, r_{e_s}, r_{k_1}, \ldots, r_{k_l}) = \mathcal{F}_e^{t_*-1}(r_{e_1}, \ldots, r_{e_s}, r_{k_1}, \ldots, r_{k_l})$$

then we have that for any $e \in E$

$$\begin{aligned}
\alpha'_1(e) &= \pi'_e(\sigma'(e_1, e) \cdot \alpha'_0(e_1), \ldots, \sigma'(e_s, e) \cdot \alpha'_0(e_s), \sigma'(k_1, e) \cdot \alpha'_0(k_1), \ldots, \sigma'(k_l, e) \cdot \alpha'_0(k_l)) \\
&= \pi'_e(\alpha'_0(e_1), \ldots, \alpha'_0(e_s), \alpha'_0(k_1), \ldots, \alpha'_0(k_l)) \\
&= \pi'_e(\alpha_0(e_1), \ldots, \alpha_0(e_s), \alpha_0(k_1), \ldots, \alpha_0(k_l)) \\
&= \mathcal{F}_e^{t_*-1}(\alpha_0(e_1), \ldots, \alpha_0(e_s), \alpha_0(k_1), \ldots, \alpha_0(k_l)) \\
&= \alpha_{t_*-1}(e)
\end{aligned}$$

Consequently, the activation of all IE nodes in N' after similarity propagation will be the same as in N after time $t_* - 1$ and, thus all case nodes will achieve the same degree of activation in the following step. ◇

In the proof of Theorem 6.2, a BCRN has been constructed which is equivalent to a given CCRN. In particular, very much knowledge has been put into the propagation functions, i.e. the entire similarity knowledge contained in the relationships between IE and concept nodes has

been *compiled* and, thus, a specific propagation function for each node has been constructed. The similarity arcs, on the other hand, do no longer contain any valuable knowledge about similarities. This was necessary to capture the general case in which an arbitrary similarity measure can be implemented by means of a Case Retrieval Net. Similarly to the relationships between Theorems 5.3 and 5.2, the above compilation process is much simpler if a composite similarity measure is used:

Theorem 6.3 *For any given CCRN $N = [E, C, K, \sigma, \rho, \delta, \Pi, t_*]$ implementing a composite similarity measure, an equivalent BCRN can be constructed by adjusting the similarity arcs adequately.*

Proof:
Let $N' = [E', C', \sigma', \rho', \Pi']$ be a BCRN with

- $E' = E \cup K$, $E' = \{e_1, \ldots, e_s, k_1, \ldots, k_l\}$
- $C' = C$
- $\sigma' : E' \times E' \to \mathcal{R}$ such that

$$\forall n_i, n_j \in E' \sigma'(n_i, n_j) = \mathcal{F}_{n_i}^{t_*-1}(0, \ldots, 0, 1, 0, \ldots, 0)$$

  with 1 being on the position u such that

$$u = \begin{cases} w & : \quad n_j = e_w \\ s + w & : \quad n_j = k_w \end{cases}$$

- $\rho' : E' \times C' \to \mathcal{R}$ with

$$\forall n \in E' \forall c \in C' \rho'(n, c) = \begin{cases} \rho(n, c) & : \quad n \in E \\ 0 & : \quad n \notin E \end{cases}$$

- $\forall n \in E' \pi'_n : \mathcal{R}^{E'} \to \mathcal{R}$ such that

$$\forall n \in E' \pi'_n(r_{e_1}, \ldots, r_{e_s}, r_{k_1}, \ldots, r_{k_l}) = max\{r_{e_1}, \ldots, r_{e_s}, r_{k_1}, \ldots, r_{k_l}\}$$

- $\forall c \in C' \pi'_c : \mathcal{R}^{E'} \to \mathcal{R}$ such that the propagation functions of case nodes are independent of the activation of formerly concept nodes:

$$\pi'_c(r_{e_1}, \ldots, r_{e_s}, r_{k_1}, \ldots, r_{k_l}) = \pi_c(r_{e_1}, \ldots, r_{e_s})$$

Furthermore, initial activation will be performed as in BCRNs depending on the query $q = [q^1, \ldots, q^n]$:

$$\alpha'_0(c) = 0, c \in C'$$

and

$$\alpha'_0(e) = \begin{cases} 1 & : \quad e \in \{q^1, \ldots, q^n\} \\ 0 & : \quad e \notin \{q^1, \ldots, q^n\} \end{cases}$$

When considering the activation of IE nodes after similarity propagation in this BCRN, we have for an arbitrary $e \in E$ with $type(e) = A_i$

$$\begin{aligned}
\alpha'_1(e) &= \pi'_e(\sigma'(e_1, e) \cdot \alpha'_0(e_1), \ldots, \sigma'(e_s, e) \cdot \alpha'_0(e_s), \sigma'(k_1, e) \cdot \alpha'_0(k_1), \ldots, \sigma'(k_l, e) \cdot \alpha'_0(k_l)) \\
&= max\{\sigma'(e_1, e) \cdot \alpha'_0(e_1), \ldots, \sigma'(e_s, e) \cdot \alpha'_0(e_s), \sigma'(k_1, e) \cdot \alpha'_0(k_1), \ldots, \sigma'(k_l, e) \cdot \alpha'_0(k_l)\} \\
&= max\{\sigma'(q^1, e) \cdot \alpha'_0(q^1), \ldots, \sigma'(q^n, e) \cdot \alpha'_0(q^n)\} \\
&= max\{\sigma'(q^1, e), \ldots, \sigma'(q^n, e)\} \\
&= \sigma'(q^i, e) \\
&= \mathcal{F}_e^{t_*-1}(0, \ldots, 0, 1, 0, \ldots, 0) \\
&= \alpha_1(e)
\end{aligned}$$

$\diamond$

Therefore, if a a composite similarity measure is being used, a CCRN can be compiled to a much simpler BCRN than has been possible for the general case. In fact, we do not really need a composite similarity measure but only that for each IE node e there is at most one element in the query which influences the activation of e. Figuratively speaking, *interferences* can be ruled out which might prevent us from determining appropriate degrees of similarity between IE nodes because any such similarity might depend on the activation of yet other IE nodes.

6.2 Microfeature Case Retrieval Nets

In this section, we will discuss a Case Retrieval Net model that is formally related to Conceptual CRNs but provides for a slightly different interpretation. Again the motivation for this type of model is that the similarity between IEs might be better representable by a certain structure rather than as an explicit link.

The idea of using so-called *microfeatures* is based on the observation that in many application domains the features describing a case can be grouped in three categories:

Features with functionally expressible similarity relationships:
For a number of features, fairly simple and straightforward functions can be specified which can be used to compute the similarity between arbitrary values of that feature. For numerical attributes, for example, a kind of inverse distance measure can often be utilized.

Features with extensionally expressible similarity relationships:
In particular for attributes with fairly few values it is often possible to relate these values explicitly. Basically, a kind of table can be filled in which, for all possible pairs of attribute values, the corresponding value of σ is specified.

Features with intensionally representable similarity relationships:
Often features exist the values of which have a *deeper* meaning. That is, a single value of such an attribute is, in fact, an abstraction for a number of properties, or *microfeatures*, of that case.

Example 6.1 In the VIRTUAL TRAVEL AGENCY, examples for these three feature types are (in that order) the `Price`, the `Comfort`, and the `Destination`. For the latter it is, in principle, possible to explicitly specify a similarity matrix such that the similarity function could also be described extensionally. However, it seems more adequate to select alternative traveling destinations because of similar properties, such as the `landscape`, the `culture`, the `climate`, and available `leisure time facilities`.

6.2.1 Model-Based Similarity Assessment

The goal of Microfeature Case Retrieval Nets (MFCRNs) is to utilize existing background knowledge in order to describe features of the latter kind. The degree of similarity between two attribute values is based on the number of microfeatures common to both values (cf. also Sutcliffe 1992). In a certain sense, a (fairly simple) domain model is used to assess the similarity of IEs. Figure 6.2 sketches a part of a MFCRN for the VIRTUAL TRAVEL AGENCY domain.

Formally, MFCRNs are CCRNs with some specific properties:

- There is just one level of concept nodes which are referred to as microfeature nodes here.
- The propagation process is performed in 4 steps, i.e. the activations $\alpha_3(c), c \in C$ are used for establishing the preference ordering.

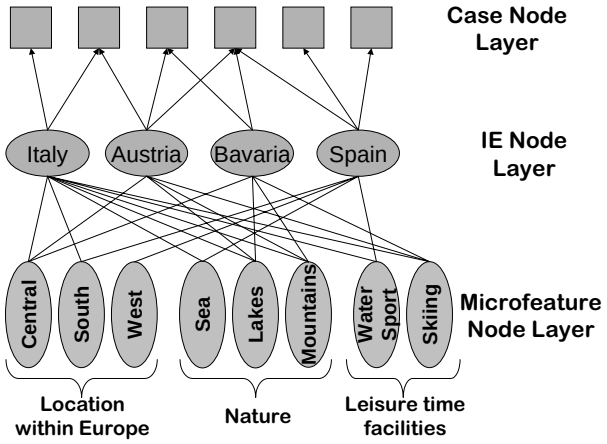


Figure 6.2: Example of a Microfeature CRN for the VIRTUAL TRAVEL AGENCY domain: The various IEs representing possible `Destination`s can be compared in terms of properties on a more detailed level. Other features are not shown here.

- the prototypicality function δ has the semantics of
  - `has-property` if it describes the relation between an IE and a microfeature;
  - `is-property-of` if it describes the relation between a microfeature and an IE;
  - similarity of microfeatures otherwise.

As a consequence of the above relationships between MFCRNs and CCRNs, it is not necessary to investigate the equivalence of BCRNs and the possible *compilation* of MFCRNs to BCRNs. Instead, we will focus here on interesting properties of MFCRNs which, however, would require further analysis for implementation.

6.2.2 Context-Dependent Similarity Assessment

While the initial idea was to utilize MFCRNs in domains with composite similarity measures in order to describe the similarity of IEs representing values of specific types of features, a composite similarity measure is not really required from a formal point of view. Rather, microfeatures might as well describe relationships between IEs belonging to *different* features.

> **Example 6.2** In the VIRTUAL TRAVEL AGENCY domain, a query by a customer might express, apart from the `Destination` a certain `Activity` or sport the customer wants to perform. Given this, some microfeatures initially intended for describing the `Destination` IEs might be weighted such that they also support the `Activity` and, thus, influence which other `Destination`s are the best alternatives. Given the net sketched in Figure 6.2, `Spain` might be the best alternative to `Italy` if the customer wants to go sailing — while `Austria` is a better choice for skiing.

In this way, dependencies among the features can be represented via microfeatures which could not be modeled by a composite similarity measure. On the other hand, the overall similarity measure still may very much rely on comparing local similarity values for single features and averaging these local ones to obtain a global similarity measure. In that sense, the similarity measure could be considered as *nearly composite*. However, having these kind of dependencies between different features prevents us from being able to *compile* a MFCRN into a BCRN in the simple way Theorem 6.3 suggested, i.e. by determining appropriate values for the similarity function σ.

6.2.3 Learning in MFCRNs

Since Microfeature CRNs, as CRNs in general, are somewhat similar to Artificial Neural Networks (ANNs), it is worth exploring some ANN–like learning approaches to train or adjust MFCRNs. A major difference between ANNs and MFCRNs, however, is that in the latter the meaning of *hidden* units (i.e. microfeatures) is known whereas this is typically not true for ANNs. Hence, this knowledge should be utilized while training the net.

When considering the structure of MFCRNs, learning seems promising with respect to three different parameters:

$\delta(e,k)$ for $e \in E, k \in K$, which expresses how strong a particular IE is associated to a microfeature;

$\delta(k,e)$ for $k \in K, e \in E$, which describes the *importance* of a microfeature for an IE;

$\delta(k_1,k_2)$ for $k_1, k_2 \in K$, which represents the similarity of microfeatures.

Really applying learning techniques in MFCRNs would definitely require further work. We only performed some initial experiments in which we assumed that learning $\delta(k_1,k_2)$ would not be appropriate because these values are often straightforward to determine.

Instead, we tested some learning techniques known from ANNs (such as delta rule learning) in order to learn the weights of the connections between IEs and microfeatures (in both directions). This has the major advantage that feedback provided by the user could be analyzed by the system:

- If an IE has been suggested by the system as the best alternative and the user accepted the system's suggestion, then the arcs from microfeatures to that IE would be strengthened.
- If, however, the most active IE was rejected by the user, then these connections would be weakened.

While some initial tests with this type of learning have been promising, they have never really been addressed in detail. Also, these extensions have only been implemented in prototypical systems (Lenz and Burkhard 1996a).

6.3 Object-Directed CRNs

In this section, we present another extension of the Basic Case Retrieval Net model, namely *Object-Directed Case Retrieval Nets* (OCRNs), which have been developed in particular for a project in the area of technical diagnosis (Lenz, Burkhard, and Brückner 1996). The task was to develop a case-based diagnostic assistance system for a heterogeneous computer network in cooperation with *PSI* AG, Berlin. A major characteristic in this environment is that broad knowledge about the physical devices and their relationships exists and should be taken into account when performing inferences. However, this knowledge does not suffice for a pure model-based approach and, hence, we decided to apply Case-Based Reasoning but to integrate the object model during the Retrieve phase.

6.3.1 The OCRN Model

In contrast to the extensions presented in the previous sections, OCRNs are extended in so far as an additional component is placed *on top* of an existing BCRN rather than integrating new nodes and arcs directly in the net. Thus, it is crucial to note that the purpose of an OCRN is *not* to achieve an object-oriented representation of cases but to enhance the case memory with an additional structure representing relationships in the outside world. Nevertheless, the model

can formally be described as a single net. For this, we will assume that each IE $e \in E$ has an associated *type* in the sense that $\mathcal{T} = \{t_1, \ldots, t_n\}$ is a set of types and E can be partitioned according to $\mathcal{T}$:

$$E = E_1 \cup E_2 \cup \ldots \cup E_n$$

where

$$E_i \cap E_j = \emptyset \text{ for } i \neq j$$

We will further assume a function

$$type : E \to \mathcal{T}$$

such that $type(e_i) = t_j$ denotes that e_i is of type t_j and that any case may have at most one IE per type.

Definition 6.5 (Object-Directed Case Retrieval Net) *An* Object-Directed Case Retrieval Net *(OCRN) is defined as a structure* $N = [E, C, \sigma, \rho, \Pi, O, \psi, \eta]$ *such that*

$[E, C, \sigma, \rho, \Pi]$ *is a BCRN where each IE* $e \in E$ *is assumed to represent a specific symptom in the application as an attribute–value pair;*

O *is the finite set of* object nodes *where each object* o *is a pair* (id_o, E_o) *consisting of a unique name* id_o *and an associated set of IEs* $E_o \subseteq E$

η *is the* property *function*

$$\eta : E \to \mathcal{P}(\mathcal{T})$$

describing which properties are associated to the objects in the domain:

$$t \in \eta(o) \Longleftrightarrow o = (id, E_o) \wedge \exists e \in E_o : type(e) = t$$

we will refer to this as o *having the property* t*;*

ψ *is the* hierarchy *function relating the objects in the domain into an object hierarchy*

$$\psi : O \times O \to \{0, 1\}$$

such that

$$\psi(o_1, o_2) = 1 \Longleftrightarrow (o_1 \textit{ is a } o_2) \vee (o_1 \textit{ is part of } o_2)$$

□

6.3.2 Retrieval in OCRNs

Similarly to the retrieval process in BCRNs, case retrieval in CCRNs is performed by propagating activations through the net. During this propagation process, activation functions of IE nodes as well as case nodes are considered:

Definition 6.6 (Activation of a OCRN) *An* activation *of an Object-Directed CRN* $N = [E, C, \sigma, \rho, \Pi, O, \psi, \eta]$ *is a function*

$$\alpha : E \cup C \to \mathcal{R}$$

□

Exactly as for BCRNs (cf. Definition 5.2), these *activations* are annotations to the nodes. The actual value of activation of each node depends on time and can be computed as defined for BCRNs in Definition 5.3 on page 44.

For the object nodes, the situation is different. The idea here is, for a given activation of IE and case nodes, to determine an object which according to its properties allows to differentiate

the set of possible diagnostic hypotheses. More precisely, if after the usual propagation process a number of cases are highly activated which suggest different diagnosis, then the object model is used to determine (a) which objects might help in differentiating the cases and (b) which object is the most specific of that kind. Given this, information on the properties of that object are requested (e.g. from the user) and the propagation process is repeated with the query being extended by the information obtained that way. The following pseudo-code further illustrates this process.

```
BEGIN PROCEDURE RetrieveOCRN([q^1,...,q^n], k)
  Let q = {q^1,...,q^n}
  Initially activate IEs for q as in BCRNs                    /*1*/
  Propagate similarity within E as in BCRNs
  Propagate relevance to case nodes as in BCRNs
  Let C_Hyp be the set of highest activated cases             /*2*/
  IF all cases in C_Hyp share the diagnosis d
    // No further diagnostic process required
    EXIT with diagnosis d
  ENDIF
  // Determine which IEs help to
  // further to distinguish diagnoses
  Let E_Rel = {e ∈ E|∃c ∈ C_Hyp : ρ(e,c) > 0 ∧ α_2(e) = 0}    /*3*/
  // Determine the set of objects that can contribute
  // further information to the relevant IEs
  Let O_Rel = {o ∈ O|∃e ∈ E_Rel : o has property type(e)}
  Let o_* be the most specific object in O_Rel wrt. ψ         /*4*/
  Let E_Useful = {e ∈ E|e ∉ q ∧ η(e,o_*) = 1}
  Let E_MostUseful = {e ∈ E_Useful that best differentiate C_Hyp}  /*5*/
  Request info about the IEs in E_MostUseful from the user
  Let q = q ∪ E_MostUseful
  Restart the entire process in step /*1*/
END PROCEDURE
```

If in step `/*4*/` multiple objects exist which are incomparable with respect to ψ, then one of the most specific objects is randomly selected. In step `/*5*/`, information–theoretic measures may be used in order to determine which IEs will provide the most information and, thus, will be most useful for establishing the diagnosis. This process is also sketched in Figure 6.3 for a very simple OCRN.

6.4 Lazy Propagation of Similarity

In BCRNs, retrieval is performed using three independent and sequentially separated steps:

1. Activation of all IEs specified for the query;
2. Spreading activation via the similarity arcs to access similar IEs;
3. Spreading activation along the relevance arcs in order to access the case nodes associated to the activated IEs.

This implies that the entire set of IEs connected by similarity arcs to some initially specified IE has to be processed before any case can be retrieved. While this approach allows for a formal investigation of the retrieval process in CRNs, situations may occur where the spread of activation should be limited to a certain degree because of two reasons:

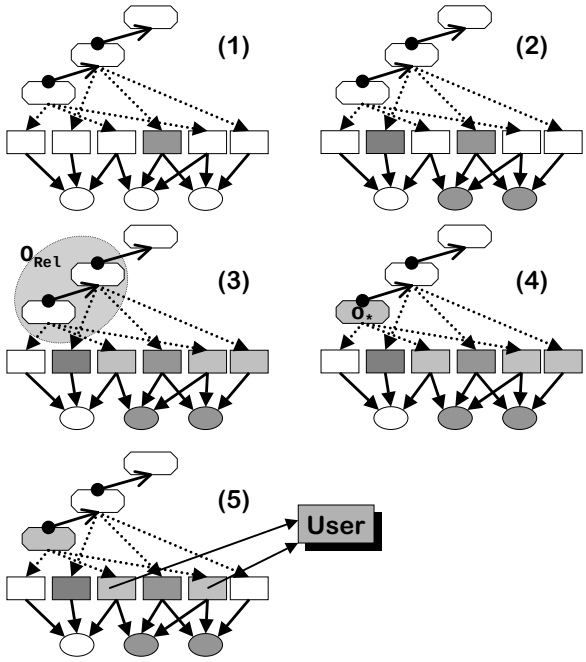


Figure 6.3: Retrieval in Object-Directed Case Retrieval Nets: After a spreading activation process as in BCRNs (step 1), the hypothetical diagnosis are collected (step 2) and relevant IEs are determined (step 3). Then, the object model is used to determine the object (step 4) the properties of which may be used to distinguish the hypothetical diagnoses. Given the most discriminating properties (step 5), information about these is requested from the user and a new retrieval process is started.

Firstly, this procedure entails that generally every IE having non-zero similarity to any part of the query will receive at least a small amount of activation, and consequently every case associated with any such IE will be considered, too[1]. Thus, even very small degrees of activation will be propagated through the net of IEs despite the fact that these might never influence the result of the retrieval process.

Secondly, it implies that the entire net is kept in memory. In particular, not only all IEs have to be explicitly represented (except when using computational nodes, as described in Section 5.5.2), but even every case existing in the application has to be represented by a case node in memory. Hence, for large case bases problems may arise simply because of a lack of memory.

Although in the applications we have dealt with we never encountered any problem with the basic retrieval procedure, we will in the following present an alternative method which also shows properties of *any-time* algorithms: Whenever the retrieval is aborted, the system is able to deliver a set of highly similar cases — but it cannot be guaranteed that these are the most similar ones. Furthermore, these techniques provide mechanisms for dynamically loading the required case nodes as will be described below.

[1]Of course, one could argue that during the design of a system one could define a similarity function that avoids this, for example by using some pruning mechanism. In general, however, the effect might still occur.

6.4.1 An Example Domain

For illustration, we will in the following utilize the fairly simple case base sketched in Table 6.1 that might, for example, list a number of available hotels, including their `Price`, the `Region` where they are situated, and the `Comfort` class they offer.

Table 6.1: Sample case base used in the following for illustrating the approach of Lazy Propagation of Similarity

Case	`Price`	`Region`	`Comfort`
c_1	DM 100	Bavaria	***
c_2	DM 95	Bavaria	**
c_3	DM 110	Austria	***
c_4	DM 95	Switzerland	*
c_5	DM 130	Italy	****
c_6	DM 150	Italy	Luxus
c_7	DM 150	South Tyrol	*****

Based on this case base, we will have 5 IEs for the feature A_1 (`Price`), 5 IEs for A_2 (`Region`), and 6 IEs for A_3 (`Comfort`). Furthermore, we will assume that the goal is to retrieve the 3 most similar cases for the query

`Price=100` and `Region=Bavaria` and `Comfort=***`

Finally, we suppose that a composite similarity measure is used and that the *local* similarities satisfy the following (quite natural) criteria:

- for `Price`:

$$\begin{aligned} sim_{Price}(100,100) &> sim_{Price}(100,95) \\ &> sim_{Price}(100,110) \\ &> sim_{Price}(100,130) \\ &> sim_{Price}(100,150) \end{aligned}$$

- for `Region`:

$$\begin{aligned} sim_{Region}(\texttt{Bavaria},\texttt{Bavaria}) &> sim_{Region}(\texttt{Bavaria},\texttt{Austria}) \\ &> sim_{Region}(\texttt{Bavaria},\texttt{Switzerland}) \\ &> sim_{Region}(\texttt{Bavaria},\texttt{SouthTyrol}) \\ &> sim_{Region}(\texttt{Bavaria},\texttt{Italy}) \end{aligned}$$

- for `Comfort`

$$\begin{aligned} sim_{Comfort}(***,***) &> sim_{Comfort}(***,****) \\ &= sim_{Comfort}(***,**) \\ &> sim_{Comfort}(***,*****) \\ &> sim_{Comfort}(***,\texttt{Luxus}) \\ &> sim_{Comfort}(***,*) \end{aligned}$$

6.4.2 Heuristic Restrictions

A simple modification to the basic retrieval method presented in Section 5.2 is to heuristically restrict the number of IEs considered during retrieval. The heuristics would be that, given an arbitrary query, only a certain variation will be accepted concerning each feature and, thus, IEs outside of this variation may be ignored during similarity propagation.

> **Example 6.3** For the `Price` feature, for instance, the available budget may simply not permit booking a hotel for `DM 130` or even `DM 150` despite the similarity function may indicate a non-zero similarity (for example when implemented by means of an inverse distance function). Hence, for the above described query the spread of activation might be restricted to the `Price` range of `DM 95` to `DM 110`.

In the general case, a *threshold* value $\varepsilon > 0$ may be inserted such that activations will only be propagated along similarity arcs if this threshold is exceeded. This method has close relationships to *relational retrieval* as considered by Wess (1995, pp. 165). In particular, it has the advantage that it may be implemented as an extension to relational databases: Instead of posing a simple database query representing the query, a disjunctive query is constructed containing exactly those IEs having a similarity higher than ε (see also Section 10.1).

A major disadvantage of using heuristic restrictions is, however, that this retrieval method cannot be guaranteed to be complete: Even if a case perfectly matches in nearly all features, a single feature value (represented by an IE) may suffice to discard that case if it causes the local similarity concerning that feature to be below the threshold ε. In the global similarity function that case could, however, still be highly similar to the query (see also Wess 1995 for a discussion of the properties of relational retrieval).

Nevertheless, we think that this type of retrieval could be applied in certain circumstances. In particular, it is often the case that very low degrees of similarity are caused by a similarity function on the representational level SIM_{Rep}, such as when using an inverse distance function in the above example. Very often, however, this similarity function will only be a kind of approximation for the actual similarity on the application level SIM_{App} which may, indeed, indicate that a price of `DM 150` may not be *acceptable* when having only a `DM 100` budget. In such situations, the pretended retrieval error can be safely ignored and heuristic restrictions may be applied thus further improving the retrieval performance.

Also, the retrieval error could be avoided if not just a single relational retrieval is performed but a sequence each with a lower threshold. This process could continue until a satisfactory result has been achieved. In fact, this closely resembles what will be explained in the following as *lazy propagation*.

6.4.3 The Idea of Lazy Propagation

An alternative to pruning certain similarity arcs (as indicated by the ε-threshold above), is to propagate along these arcs but only after the more promising paths in the net have been followed. The idea here is to introduce a kind of *laziness* into the spreading activation process: Propagation within the net of all IEs proceeds in several steps, subsequently extending the scope from more similar values to less similar ones as illustrated in Figure 6.4. At the same time, the cases connected to these IEs are activated (Lenz and Burkhard 1996b). Thus, it is possible to prune the process of spreading activation (and omit the treatment of highly dissimilar values) as soon as a sufficiently large number of cases has been activated during this process.

Thus *laziness* refers to the way how more and more IEs are taken into account during similarity activation over time. In some sense, this is related to a search method known in AI named *iterative deepening* in that a single search step is performed, then the success criterion is checked and if that fails the process continues.

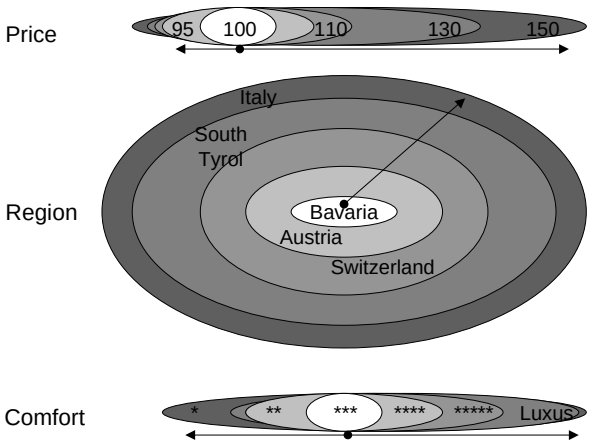


Figure 6.4: Basic idea of *Lazy Propagation of Similarity* in the space of all IEs: After the IEs specified in the query have been initially activated, the retrieval is performed by subsequently considering less similar IEs (indicated by darker shading) and activating the cases associated to these (not shown here). The arrows indicate decreasing local similarities to the IE expressed in the query.

Example 6.4 For the hotel case base given in Table 6.1 and the query listed above, retrieval might be performed by

1. initially activating the IEs representing `Price=100`, `Region=Bavaria`, and `Comfort=***`
2. spreading activation along the relevance arcs starting in these 3 IEs (thus accessing c_1, c_2, and c_3, but the latter two only for a subset of all features)
3. spreading activation along the similarity arcs to `Price=95`, `Region=Austria`, `Comfort=**`, and `Comfort=****`
4. spreading activation along the relevance arcs starting in these 3 IEs (thus accessing c_4 and c_5, but again only for a subset of all features)
5. spreading activation along the similarity arcs to `Price=110`, `Region=Switzerland`, and `Comfort=*****`
6. spreading activation along the relevance arcs starting in these 3 IEs (thus completing the activation of c_1, c_2, and c_3)
7. ...

Figure 6.4 illustrates this process. which is continued until a sufficient number of cases has been found, i.e. completely activated concerning all features (as is the case here after step 6).

However, in this very basic form, case retrieval would suffer from the same drawbacks as did the heuristic restrictions, namely that completeness and correctness of retrieval can not be guaranteed. Rather, it is well possible that a case having received input from all attributes is less similar to the initial query than a case being only partially activated.

In the above example, c_3 is completely activated after step 6 while c_4 is not. But since c_4 is concerning the `Price` feature much closer to the query, it might be more similar than c_3 if

the features are appropriately weighted. Nevertheless, the above idea may be extended into a procedure that allows for a complete and correct retrieval procedure. This formalism will be given in the following.

6.4.4 Formal Requirements

In order to discuss the retrieval method of *Lazy Propagation of Similarity* (LPS), we will in the following abstract from a specific similarity measure and instead consider a preference ordering of the case base CB. In Section 6.4.8 it is shown how this scheme can be converted to an actual similarity measure.

The Query Space

Definition 6.7 (Query Space) *The* query space QS *is the set of all possible queries that can be posed to a CBR system:*

$$QS = \mathcal{P}(E)$$

□

While in the general framework of CRNs a query can be any subset of IEs (i.e. QS is the power set of E), it may be re-written in case of composite similarity measures as:

$$QS = \mathsf{domain}(A_1) \times \ldots \times \mathsf{domain}(A_k)$$

where $dom(A_i)$ is the domain of the ith attribute A_i.

Preference Orderings

Definition 6.8 (Global preference ordering) *A* global preference ordering Pre *is a subset from* $QS \times CB \times CB$ *with the following semantics:*

$$Pre(q, x, y) \Longleftrightarrow x \textit{ is at least as similar to } q \textit{ as } y \textit{ is}$$

□

Pre thus induces a partial ordering $\succeq^q$ of the cases in CB according to their similarity to a specific query q:

$$Pre(q, x, y) \Longleftrightarrow x \succeq^q y$$

This ordering is called *global* as it considers the ordering of the entire cases, i.e. from a *holistic* point of view. In the following we will assume, that a specific query q has been selected and omit the index. Also, when comparative terms (e.g. *worse* or *better*) are used in the following, these always refer to the preference ordering Pre.

Definition 6.9 (Local preference ordering) *A* local preference ordering Pre_{A_i} *is similar to* Pre *except that for establishing the ordering of cases only the information concerning the specific attribute* A_i *is taken into account:*

$$Pre_{A_i}(q, x, y) \Longleftrightarrow \textit{ concerning attribute } A_i, \textit{ } x \textit{ is at least as similar to } q \textit{ as } y \textit{ is}$$

□

Pre_{A_i}, too, induces a partial ordering $\succeq^q_{A_i}$ of the cases according to their similarity to a specific query q for a particular attribute A_i:

$$Pre_{A_i}(q, x, y) \Longleftrightarrow x \succeq^q_{A_i} y$$

However, it may well be that different local preference orderings result from a set of attributes whereas there will be just one global one. As above, the index q will be omitted in the following.

Requirements on Similarity Measures

Finally, we assume that an arbitrary similarity measure is used to compare cases and queries which, however, satisfies the following requirements:

Requirement 1: The similarity measure applied is composite.

Requirement 2: The global preference ordering Pre induced by the similarity measure is monotonous for each attribute:

$$\forall c_i, c_j \in CB : ((\forall A_i : c_i \succeq_{A_i} c_j) \Longrightarrow c_i \succeq c_j)$$

i.e. if a case c_i is better (in terms of all local preference orderings) than a case c_j, then c_j cannot be better than c_i (in terms of the global preference ordering).

Requirement 3 The goal of the Retrieve phase is to find the k cases most similar to the query. (Alternatively, the goal might be to retrieve cases being more similar than a given threshold. Then, the formalism below would have to be adapted accordingly.)

The Lazy Spreading Activation Process

Compared to the 3–step retrieval process in BCRNs described on page 44, the activation if IEs and case nodes will be computed by taking the feature similarities into account:

Definition 6.10 (Lazy spreading activation process) *Let $N = [E, C, \sigma, \rho, \Pi]$ be a BCRN which implements a composite similarity function (thus, $E = \{e_1, \ldots, e_s\} = E_{A_1} \cup \ldots \cup E_{A_n}$). Further assume that retrieval should be performed for a query $q = [q^1, \ldots, q^n]$. The activation of the nodes in N at time t is computed by an activation function*

$$\alpha_t : E \cup C \rightarrow \mathcal{R}$$

as follows:

Initial Activation:

$$\alpha_0(e) = \begin{cases} 1 & : \quad e \in \{q^1, \ldots, q^n\} \\ 0 & : \quad e \in E \text{ otherwise} \end{cases}$$

$$\alpha_0(c) = 0, c \in C$$

Lazy Spreading Activation: *Activation is propagated to only those IEs which belong to the group of most similar IEs to the given query; the size of this group is extended over time:*

$$\alpha_{t+1}(e) = \begin{cases} \pi_e(\sigma(e_1, e) \cdot \alpha_t(e_1), \ldots, \sigma(e_s, e) \cdot \alpha_t(e_s)) \\ \qquad \text{if } type(e) = A_i \text{ and } e \text{ belongs to } t+1 \text{ most similar IEs to } q^i \\ 0 \qquad e \in E \text{ else} \end{cases}$$

Relevance Propagation: *After each lazy spreading activation step, activation is propagated to the case nodes $c \in C$:*

$$\alpha_{t+1}(c) = \pi_c(\rho(e_1, c) \cdot \alpha_t(e_1), \ldots, \rho(e_s, c) \cdot \alpha_t(e_s))$$

Note that this formal definition seems to be highly complicated. In Section 6.4.8 a straightforward implementation will be described which realizes the above activation scheme by ordering the similarity arcs of IE nodes according to their associated weights.

To describe the *Lazy Spreading Activation* process, we furthermore need to identify those IEs which have been activated so far, i.e. to which the spreading activation process has been performed.

Definition 6.11 (Active set of IEs) *At any stage of retrieval, the* active set *of IEs (AS) refers to the set of all IEs currently within the scope of considered local preference orderings:*

$$AS(t) = \{e \in E : \alpha_t(e) \neq 0\}$$

□

In Figure 6.4, for example, the AS's at different stages of retrieval are marked by the shaded areas, where darker shading implies that the IEs are activated at a later stage of the propagation process.

As a consequence of only activating subsets of IEs at any given time, three different sets of case nodes may be distinguished:

Definition 6.12 (Partitions of case nodes) *The sets M_C, M_P and M_N are defined as follows:*

M_C *contains all* completely activated cases, *i.e. those cases all associated IEs of which are already in the current AS:*

$$M_C(t) = \{c \in C : c = \{c^1, \ldots, c^n\} \wedge \forall c^i : c^i \in AS(t)\}$$

M_P *contains all* partially activated cases*:*

$$M_P(t) = \{c \in C : c = \{c^1, \ldots, c^n\} \wedge c \notin M_C(t) \wedge \exists c^i : c^i \in AS(t)\}$$

M_N *contains all* not activated cases*:*

$$M_N(t) = \{c \in C : c \notin (M_C(t) \cup M_P(t))\}$$

□

Obviously, at any given time t, $M_C(t)$, $M_P(t)$, and $M_N(t)$ form a partition of the case base: At any stage of retrieval, each case is element of exactly one set. This will be utilized in the following.

α and β Retrieval Errors

For a better understanding of the techniques presented below, a classification of retrieval errors presented by Wess (1995, p. 167) is highly useful. We will not go into formal properties of retrieval methods in general and possible retrieval errors in particular. Rather, Figure 6.5 should be sufficient for illustrating the two types of errors referred to as α– and β–error, respectively:

α–error: the retrieval method has excluded cases despite them being sufficiently similar to the query;

β–error: the retrieval method has considered cases which appear to be too dissimilar to the query.

In a nutshell, an α–error indicates that case retrieval is incomplete while β–errors indicate that this process is inefficient. Of course, any four combinations are possible, as displayed in Figure 6.5:

(a) Case retrieval is efficient as well as complete.

(b) Case retrieval is efficient but incomplete.

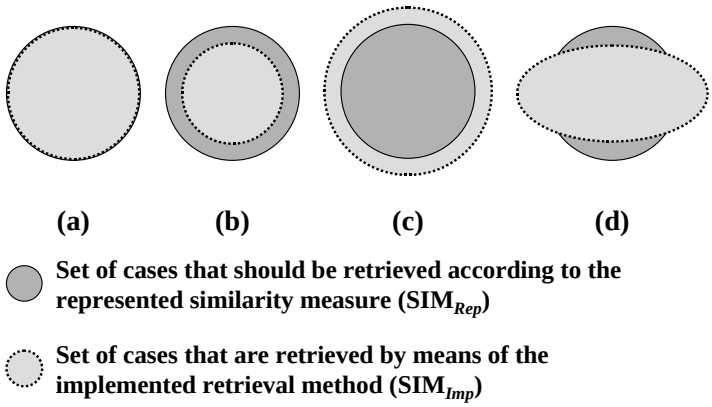


Figure 6.5: Illustration of possible retrieval errors: The query is located in the center of the regions, the dark circle to the represented similarity, and the dotted circle to the similarity on the implementation level.

(c) Case retrieval is inefficient but complete.

(d) Case retrieval is both inefficient and incomplete.

Concerning the β–error it is important to take all cases into account which the retrieval method did somehow consider in detail. For example, a linear search through case memory might return exactly the k most similar cases — but on the expense of having computed the similarity of each single case in case memory to the given query. This, obviously, is a tremendous β–error.

6.4.5 Improvement 1: Avoiding α–Errors

Given the above formalism and a query $q = [q^1, \ldots, q^n]$ for which the k best cases should be retrieved, correctness and completeness of the *lazy spreading activation* approach can be achieved when the retrieval process itself is implemented in two subsequent phases:

Lazy Propagation: performing the process as described in Definition 6.10 until either all IEs have been activated or the active set of IEs AS contains at least k cases

Error Avoidance: investigating the cases contained in M_P after the previous phase has been finished

The following pseudo-code listing shows how this method might be implemented. As Theorem 6.4 below will show, limiting the second phase to only those cases which are partially activated (marked `/* 1 */`) will improve efficiency while still guaranteeing correctness of retrieval in the sense that the retrieval process will, indeed, deliver the set of most similar cases.

```
BEGIN PROCEDURE LazyPropagation1([q^1,...,q^n], k)
  Let t = 0
  AS(0) = {q^1,...,q^n}
  M_C(t) = ∅
  M_P(t) = ∅
  WHILE |M_C(t)| < k
    t := t + 1
    determine AS(t)
    determine M_P(t)
    determine M_C(t)
  END /* WHILE */
  FOR EACH case c_i ∈ M_P(t)                                    /* 1 */
    compute SIM([q^1,...,q^n], c_i)
    insert c_i into M_C(t)
  END /* FOR */
  RETURN the k best cases from M_C(t)
END /* PROCEDURE */
```

Example 6.5 Recall the sample case base and the assumed similarity functions given in Section 6.4.1. The IEs that are in AS as well as the partitioning of the case base after each step are listed in Table 6.2 on page 81.

Table 6.2: Illustration of the lazy spreading activation approach for the sample case base.

t	$AS(t)$	M_C	M_P	M_N
1	100 Bavaria ***	$\{c_1\}$	$\{c_2, c_3\}$	$\{c_4, c_5, c_6, c_7\}$
2	95,100 Bavaria,Austria **,***,****	$\{c_1, c_2\}$	$\{c_3, c_4, c_5\}$	$\{c_6, c_7\}$
3	95,100,110 Bavaria,Austria,Switzerland **,***,****,*****	$\{c_1, c_2, c_3\}$	$\{c_4, c_5\}$	$\{c_6, c_7\}$
4	95,100,110,130 Bavaria,Austria,Switzerland South Tyrol **,***,****,*****,Luxus	$\{c_1, c_2, c_3\}$	$\{c_4, c_5, c_6, c_7\}$	$\emptyset$
5	95,100,110,130,150 Bavaria,Austria,Switzerland South Tyrol,Italy *,**,***,****,*****,Luxus	$\{c_1, \ldots, c_7\}$	$\emptyset$	$\emptyset$

For the sample case base and query, 3 completely activated cases have been found after the third step (see Table 6.2). Hence, the first phase of the lazy spreading activation procedure is finished. Note that for completely activated cases, the activation is equivalent to the similarity to the query according to Proposition 5.1.

However, terminating retrieval at this stage might result in a retrieval error similar to the retrieval with heuristic restrictions explained in Section 6.4.2. In the example, it is obvious that the cases c_6 and c_7 are worse than any of the three cases in M_C after $t = 3$. On the other hand, it is not clear at first glance how the cases in $M_P(3)$ are related to those in $M_C(3)$. For

example, c_4 appears to be closer to the query in terms of the `Price` than, say, c_3. Consequently, c_4 might also be more similar to the query in terms of the global similarity measure — e.g. if the feature `Price` is given sufficient weight.

Hence, we have to calculate the similarity to the query for the cases in $M_P(3)$ and compare the resulting values against the activations of the cases in $M_C(3)$. After that, we can

1. remove the partially activated cases from $M_P(3)$ (because explicitly computing the similarity is equivalent to completely activating cases);
2. insert those cases into $M_C(3)$; and
3. build the preference ordering of cases based on the activations of all cases in $M_P(3)$.

As a result, the three most similar cases in the example would be selected from the set $\{c_1, c_2, c_3, c_4, c_5\}$ while c_6 and c_7 could be safely ignored.

The following theorem shows that, in general, no retrieval error will occur when

Theorem 6.4 *Given a query q, a case in the case base may belong to the k best cases (according to the global preference ordering Pre) only if it is an element of $M_C \cup M_P$ when M_C contains k cases.*

Proof:
Let t_* be the time when at least k cases are completely activated for the first time:

$$t_* = min\{t : |M_C(t)| \geq k\}$$

Furthermore, let c_* be the worst case in $M_C(t_*)$:

$$c_* = c_k \in M_C(t_*) \text{ such that } \forall c_i \in M_C(t_*) : \alpha_{t_*}(c_i) \geq \alpha_{t_*}(c_*)$$

If $c = [c_1, \ldots, c_n]$ is an arbitrary case from $M_N(t_*)$, then it follows that

$$\forall A_i : c_* \succeq_{A_i} C$$

because c is not even partially activated. Due to the assumption of a monotonous preference ordering (Requirement 2) c, therefore, can not be better than c_* and, hence, does not belong to the k best cases. $\diamond$

As a consequence of Theorem 6.4, all cases which are not even partially activated can be safely ignored during the second phase of this retrieval procedure. While for the example this only concerns two cases (c_6, c_7), the savings will be much larger in real-world situations.

6.4.6 Improvement 2: Reducing β–Errors

When taking a closer look at the extended example from above, one can observe that, as a consequence of the monotonous preference ordering, the following relations must hold with respect to the global preference ordering::

- $c_1 \succeq^q \{c_2, c_3, c_4, c_5, c_6, c_7\}$
- $c_2 \succeq^q \{c_4, c_5, c_6, c_7\}$
- $c_3 \succeq^q \{c_5, c_6, c_7\}$
- $c_5 \succeq^q c_6$
- $c_7 \succeq^q c_6$

For example, $c_1 \succeq^q c_2$ holds because

$$c_1 \succeq^q_{Price} c_2 \wedge c_1 \succeq^q_{Region} c_2 \wedge c_1 \succeq^q_{Comfort} c_2$$

On the other hand, it is not clear what the result of a comparison of, for example, c_2 and c_3 will be: While c_2 is better than c_3 for the `Price` and `Region`, the reverse holds for the `Comfort`. Consequently, just the following global preference orderings of the cases are possible:

- $c_1 \succeq c_2 \succeq c_3 \succeq \{c_4, c_5, c_6, c_7\}$
- $c_1 \succeq c_2 \succeq c_4 \succeq c_3 \succeq \{c_5, c_6, c_7\}$
- $c_1 \succeq c_3 \succeq c_2 \succeq \{c_4, c_5, c_6, c_7\}$

As it turns out, c_4 is at the end of the lazy propagation phase the only partially activated case that is in at least one feature better than a completely activated case:

$$\exists A_i \exists c_j : c_j \in M_P(3) \wedge c_j \not\succeq_{A_i} c_4$$

In the example it would, therefore, be sufficient to calculate the similarity of c_4 to the query because it is guaranteed that the 3 most similar cases have to be taken from the set $\{c_1, c_2, c_3, c_4\}$.

The following theorem covers the general case and is a refinement of Theorem 6.4:

Theorem 6.5 *Given a query q, a case c_* in the case base may belong to the k best cases (according to the global preference ordering Pre) only if there is an attribute A_i and a case $c_j \in M_C$ such that $c_* \succeq_{A_i} c_j$.*

Proof:
Again, let t_* be the time when at least k cases are completely activated for the first time:

$$t_* = min\{t : |M_C(t)| \geq k\}$$

Let c be an arbitrary case such that for no case $c_i \in M_C(t_*)$ there exists an attribute A_i such that c is better in terms of the local preference ordering corresponding to A_i:

$$\forall c_j \in M_C(t_*) \forall A_i : c_j \succeq_{A_i} C$$

From the assumption of a monotonous preference ordering (Requirement 2) then follows that

$$\forall c_j \in M_C : c_j \succeq c$$

and hence that all cases in M_C are better in terms of the (global) preference ordering than c which, therefore, can not belong to the k best cases. $\diamond$

Theorem 6.5 shows that the local preference orderings concerning the employed attributes may be utilized to decrease the number of cases which have to be explored in order to avoid α–errors. By restricting this set of cases, the β–error is reduced, too.

To make use of this theorem, information about the local preference orderings for each attribute is required. This information has to be maintained in an additional data structure $M_P(A_i, t)$ and can be employed as sketched in the following listing.

```
BEGIN PROCEDURE LazyPropagation2([q^1, ..., q^n], k)
  Let t = 0
  AS(0) = {q^1, ..., q^n}
  M_C(t) = ∅
  M_P(t) = ∅
  WHILE |M_C(t)| < k
    t := t + 1
    determine AS(t)
    FOR EACH attribute A_i
      determine M_P(A_i, t)                                   /* 1 */
    END /* FOR */
    determine M_C(t)
  END /* WHILE */
  Let OpenSet = ∅
  FOR EACH attribute A_i
    Let LocallyBetterCases = {c_j ∈ M_P(A_i, t) : ∃c_k ∈ M_P(t) ∧ c_k ⋡_{A_i} c_j}
    OpenSet = OpenSet ∪ LocallyBetterCases
  END /* FOR */
  FOR EACH case c_j ∈ OpenSet                                 /* 2 */
    compute SIM([q^1, ..., q^n], c_j)
    M_C(t) = M_C(t) ∪ {c_j}
  END /* FOR */
  RETURN the k best cases from M_C(t)
END /* PROCEDURE */
```

6.4.7 Benefits for Huge Case Bases

A crucial assumption of all Case Retrieval Net models so far has been that the entire net can be kept in memory for performing the spreading activation process, including all IE and case nodes. For really huge case bases this might cause problems simply because of shortage of memory. Although in our applications we could manage up to 250,000 cases (in the VIRTUAL TRAVEL AGENCY) without any problem, there are, of course, potential applications with far more data to be handled.

For this, the above described lazy propagation is a promising alternative. Basically, the idea is to keep in memory only the complete set of IEs as well as the similarity arcs (which should always be possible). The cases, however, could be loaded dynamically from an external database thus reducing the required memory in situations with many cases. More precisely, the method could be sketched as follows:

```
BEGIN PROCEDURE LazyPropagation3(q, k)
  Let there be 0 case nodes in the net
  Perform initial activation for q
  WHILE less than k case nodes have been constructed
    Perform similarity propagation as in LazyPropagation2()
    Determine the current active set of IEs AS
    Perform an SQL query based on AS
    FOR EACH new record returned by the SQL query
      Construct a case node representing that record
    END /* FOR */
  END /* WHILE */
  Perform relevance propagation to the existing case nodes
  Reconsider in detail all partially activated cases
END /* PROCEDURE */
```

Obviously, the effort for performing this kind of retrieval is much higher than for the original procedure. In particular, the actual spreading activation process in the CRN is interlocked with database queries. Furthermore, memory has to be allocated dynamically depending on the result sets of the SQL queries. Thus, managing this technique is much more complicated. However, it provides an approach for handling data sets of arbitrary size in that only those data records are being represented in memory which appear to have some relationships (in terms of similarity) to the query. Correctness and completeness of this approach directly follow from the formalisms presented above.

6.4.8 Notes on Implementation

The formalism presented above utilizes preference orderings instead of an actual similarity measure. This has two major consequences:

1. When defining a similarity measure, weights describing the influences of the various attributes are usually incorporated. These weights do not play any role in the above formalism, in particular Theorems 6.4 and 6.5 are independent of the applied weighting scheme.

2. In terms of implementation it might be hard (or inefficient) to compute local preference orderings as these have to be inferred after a query has been specified. However, preference orderings can be *simulated* by defining the *active sets* of IEs (AS) using similarity intervals, i.e. by using a similarity threshold which is subsequently lowered. As a consequence, less similar IEs will become involved over time as has already been visualized in Figure 6.4. As similarity links to other IEs are a substantial part of Case Retrieval Nets, this information is directly available in the IE nodes.

Part III

CRNs for Building Flexible Information Systems

Chapter 7

A General Framework for Building CRN-Based Information Systems

The engineering of software gets little respect in academia where theories and proofs are king and programs are secondary.
Roger C. Schank, Communications of the ACM, 1994

In the previous chapters, we have discussed the formal model of Case Retrieval Nets, including its properties and a number of possible extensions. Now, we will address more practical aspects, namely how an information system can be built using the technique of CRNs.

This chapter describes a general framework for this, including the principle architecture, the required components, and the methodology of building such an information system. We will use examples from the Virtual Travel Agency *explained in detail in Chapter 8 throughout the chapter to illustrate the concepts introduced.*

The following two chapters will then present case studies of projects thus giving concrete examples of applying this framework.

7.1 Types of Knowledge to Reason Upon

Barletta and Klahr (1997) arranged the different types of knowledge that a knowledge-based system may have to deal with in a triangle with respect to the potential contributions to (automatic) problem solving on the one hand and to the cost of acquiring this knowledge on the other hand. This *knowledge triangle* is shown in Figure 7.1.

According to this, case bases form a separate layer in the triangle which provides more precious problem solving knowledge than, for example, standard databases or even (textual) documentations. However, these case bases are also more difficult to obtain.

7.1.1 Case Bases as View on Data

The above knowledge triangle implies that case bases are considered as a *separate* source of knowledge which exists independently of the actual data. As we shall see in the descriptions

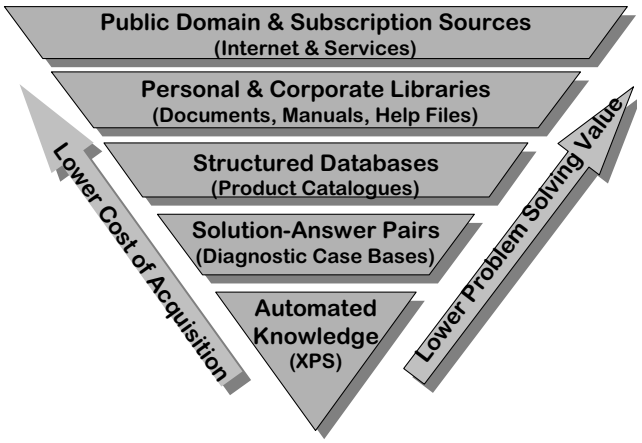


Figure 7.1: The knowledge triangle relating the various types of knowledge with respect to problem solving value and cost of acquisition (from Barletta and Klahr 1997).

of the applications in subsequent chapters, this view differs from the position taken in this thesis. An approach that seems feasible in many domains where (raw) data exists in the form of either databases or documents, is to consider the case base as a *view* on the existing data — with a notion of *view* similar to that in database technologies (Elmasri and Navathe 1994; Ullman 1988). Thus, the distinction between domain objects, cases, and indexes as described in Section 4.3 can be applied.

This can be achieved by specifying a mapping from raw data to some case format rather than performing an actual case authoring process as required for most existing CBR tools. Such a mapping will have the advantage that a case base to reason upon can be constructed automatically and that updates of the data can be mirrored in the case base without running into consistency problems. Furthermore, maintenance effort for the case base will be kept to a minimum.

On the other hand, an automatic mapping from raw data to case bases requires that this mapping can be described formally in a sufficient precise manner. In most situations, the initial effort of knowledge acquisition will, thus, be higher than for building a concrete case base. But, in principle, the same work has to be done in these situations — only on a less abstract level.

Once having implemented such a mapping, a *case base* in the traditional sense is no longer required but will only be used internally be the Case-Based Reasoning system. In the context of the Case Retrieval Net technology, this case base may even be discarded after the case memory has been constructed. (Recall that we consider a case base to be merely a set of cases whereas a case memory provides some structuring for retrieval purposes, cf. Section 2.2.)

Basically, whenever changes to the data occur, the following steps will assure that the case memory used during retrieval is up-to-date:

1. Construct a temporary case base representing the up-to-date data using the mapping procedure
2. Construct a CRN from that temporary case base
3. Discard the temporary case base as it is no longer required

Whenever data is accessed, for example for showing the contents of retrieved cases, the original data is used rather than the cases which only serve system–internal purposes. This only requires that the mapping from the data to the case base assigns unique *case descriptors* referencing the data elements.

7.1.2 Applicability

As Chapters 8 and 9 will show, the implementation of such an automatic mapping from data to case bases is feasible for many application areas. We will in particular consider situations with structured databases (for applications of electronic commerce) and with textual documents (for Textual CBR applications).

For the former, a simple *view* on the existing database(s) often suffices. That is, each record in a relational database (or a subset of it) will be considered as a case; similarly for the objects in an object–oriented database. Given this and another knowledge container for the similarity model, the Case Retrieval Net can be constructed without physically having a case base.

For Textual CBR the situation is a bit more complicated because database techniques are not applicable. As we shall see in Chapter 9, the mapping from textual documents to a case base will be performed by using a *concept dictionary* which describes how the strings occurring in a text have to be mapped to an internal case representation. Once such a dictionary exists, a kind of parser can construct the case base and the process may continue as described above.

7.1.3 The Knowledge Triangle Revisited

With this approach of seeing a case base as a view on original data, the knowledge triangle depicted in Figure 7.1 has to be modified such that the case base layer will disappear and the above two layers may become case bases by an appropriate mapping. As the following chapters will show, the Case Retrieval Net model can then be applied to both the structured database layer as well as corporate libraries. This is possible in particular because CRNs do not require cases to be solution–answer pairs (as shown in the knowledge triangle) but can handle less rigid case formats (cf. the discussion in Chapter 3).

7.2 General Components

On an abstract level, each CRN-based decision support system should consist of at least four components as shown in Figure 7.2.

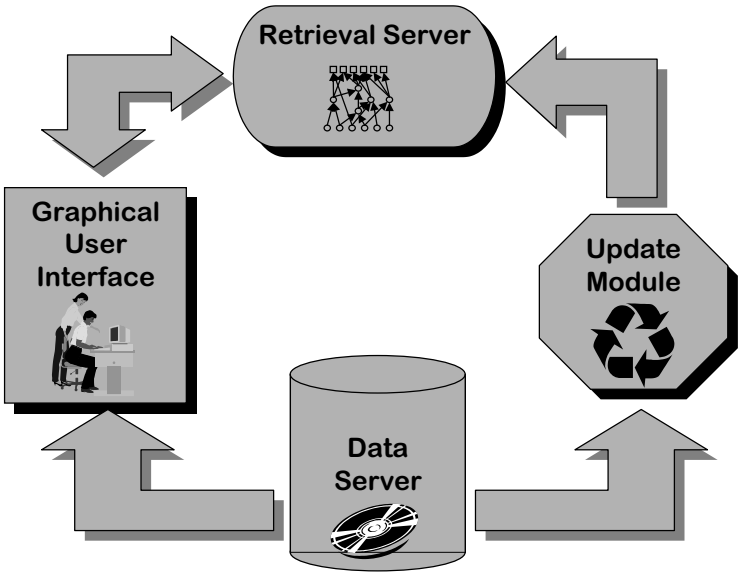


Figure 7.2: General architecture for a CRN-based decision support system

Each of these components can be considered as a *functionality module* as opposed to the *knowledge containers* discussed in Section 2.2.5. That is, each module serves a specific purpose and implements a particular functionality required within the entire system.

Each of these modules works independently of the other; well-defined communication processes specify the exchange of information between these modules. How these components interact, will also be described in Section 7.3.

Of course, every specific application may require additional components; also, under certain circumstances single components may be omitted if its functionality is either not required or performed by some other component. We will now explore the contents of each of the modules, examples and case studies will be given in the following chapters.

7.2.1 Retrieval Server

The *Retrieval Server* is the core component of every CRN-based information system. It contains the Case Retrieval Net and performs the actual retrieval process for a given query. Whether it is a true *server* in the sense of client-server architectures, depends on the complexity of the application.

In simple applications, the *Retrieval Server* may also contain other modules, for example for transforming a user-given query to the internal representation or for performing an update process. However, we suggest that — at least for more complex applications — these functionalities are provided by additional modules and the *Retrieval Server* is not overloaded with functionality beyond case retrieval.

> **Example 7.1** In an early version of the VIRTUAL TRAVEL AGENCY, for example, the *Retrieval Server* had also been used to construct the HTML page answering a user's request. This, however, caused the server to spend a large portion of performance time just for parsing templates, filling HTML forms etc. Also, the entire system slowed down due to network overload.
> Now this is avoided by delegating this task to a separate *Query and Presentation Module* (see Section 8.2.6).

7.2.2 Graphical User Interface

The *Graphical User Interface* is the set of all modules that are required for a user to communicate with the system. In general, a CRN-based information system will at least require a *Query and Presentation Module*, i.e. functionality for allowing the user to enter a query and for displaying the results of the retrieval process.

Note that in particular the design of the interface for posing a query requires careful analysis and a well-defined *Application Scenario* (see Section 7.3.1):

- The interface should, of course, be easy to use.
- What is it that a user may enter as a query: Is it a natural language description, a selection of predefined feature values, or what else?
- To what extend can the user influence the retrieval process? Can s/he give his/her own preferences or is there a static weighting applied?

> **Example 7.2** In the VIRTUAL TRAVEL AGENCY we decided to dynamically build the query interface based on the available tour packages (see Section 8.2.6). On the other hand, we did not allow for entering preferences as this would have caused an overloaded GUI.

In addition to the *Query and Presentation Module*, a *Background Info Module* will be useful in most applications. The purpose of this module is to display information beyond the pure data represented by a case. Finally, specific applications may require additional modules within the *Graphical User Interface*. This will be highly domain-dependent.

> **Example 7.3** The VIRTUAL TRAVEL AGENCY has a *Background Info Module* for displaying information about the destinations, the climate and so on. As an additional component, this application also has an *Online Booking Module* for obvious reasons.

7.2.3 Update Module

An *Update Module* is required in any information system in order to respond to changes in the underlying data. In a CRN-based decision support system, updates may be required with respect to two major knowledge containers:

- Firstly, the case data may change based on which a Case Retrieval Net has been constructed. This requires that a new CRN is constructed and the *Retrieval Server* is updated in order to use this newly built CRN.
- Secondly, the similarity knowledge may change. Since a Case Retrieval Net contains this knowledge container in a *compiled* form, the net has to be updated, too, in order to reflect the changes.

The task of the *Update Module* is to take an up-to-date case base and an up-to-date similarity measure and to construct a Case Retrieval Net. This structure may then be provided as it is to the *Retrieval Server*.

At least in complex applications, it appears to be highly useful to separate the *Update Module* from the *Retrieval Server*. By doing so,

- the update may be performed while the server is still available, this might be particularly desirable when updating is time-consuming;
- the update may even be performed on a different machine, for example because of efficiency or security reasons.

Of course, the other knowledge containers, namely the case vocabulary as well as the adaptation knowledge (see Section 2.2.5), may change, too. However, the former does, in general, not permit an automatic update whereas the latter does not influence the Retrieve phase of the CBR cycle. Consequently, the *Update Module* should primarily deal with changes in the case base and the similarity measure.

> **Example 7.4** In the VIRTUAL TRAVEL AGENCY, updates are performed automatically once new case data has arrived. The update process itself may, in peak season, require more than an hour of processing time as the data files that have to be parsed then are usually of the size of 100 to 150 MB. However, while the *Update Module* is running, the *Last Minute* application is still available due to the server being separated from the update functionality. Only after that process has been finished, the Case Retrieval Net structure is stored in a file and the server is sent an update command which causes the new CRN to be loaded.

7.2.4 Data Server

The *Data Server* is used to contain and provide access to the original data, i.e. the case data that the Case Retrieval Net had been built on, plus other data sources that provide further information but are not directly considered during retrieval, such as diagrams and pictures. It

would, in principle, be possible to also store this information directly within the *Retrieve Server*. However, this would be fairly inefficient and cause the *Retrieve Server* to be overloaded with functionality not really required for case retrieval.

Consequently, the *Data Server* is connected whenever it comes to visualize case data, for example for displaying the results of a retrieval process in a user-friendly form. Then, additional information may also be loaded from the *Data Server* which is not utilized for retrieval and, hence, not represented in the *Retrieve Server*.

Most often, the *Data Server* can be implemented either as a traditional database or via the file system.

Example 7.5 In the VIRTUAL TRAVEL AGENCY, a special module working on a specific folder in the file system serves as a *Data Server*. The files in that folder contain additional information for the case data, such as the name of the hotel, remarks (as `newly built`), or brief descriptions of the destination. This information is requested from the server when it comes to displaying the found tour packages to the user.

7.3 Building a CRN-Based Information System

After the main components of a CRN-based decision support system have been explained in the previous section, we will now describe what the usual *processes* are when it comes to really building such a system for a specific application. We will assume that the purpose of the application is to provide a service based on information disseminated by some company or organization.

Very briefly, there are four main phases addressing the following key questions:

Domain Analysis: What is the goal of the application, what data can be used for building it?

System Specification: How, precisely, can the available data be brought into a CBR system and a Case Retrieval Net?

System Implementation: How should the above discussed module be designed and implemented?

System Maintenance: Once the system has been built, how can it be updated and maintained?

Of course, these steps are well-known from standard software engineering. However, the decisions made during these steps will also influence the design of a case-based information system. For example, the targeted application scenario will strongly influence the way queries will be posed to the system, how cases can be displayed, and even whether or not cases and queries will be based on the same vocabulary. Consequently, we will discuss all these questions in detail in the following.

7.3.1 Domain Analysis

The objective of the *Domain Analysis* is to clearly specify what kind of application is required, what data this application should run on, what business processes are required in order to run the application and keep it alive, and — last but not least— who will be the typical user of the system. The *domain analysis* should be performed in cooperation by knowledge engineers, domain experts, and those later running and maintaining the system.

Domain Objects

The first questions that have to be answered refer to the data that is underlying the application:

1.1 What is the data that the application should work with; i.e. what should be the target of the the case-based search engine? What parts of the data should really go into the system?

1.2 How is this data available? Can it be obtained in an electronic format or is manual effort required? If the former, what is the format; can it be customized? If the latter, who does the job and how often is this required?

1.3 How often will the data change? Will it change in its entirety or will there be only *local* modifications?

Domain Model

Once it is clear what the domain objects are, the question arises what other components there are based on which later a similarity model might be constructed:

2.1 What knowledge sources exist within the organization that should be taken into account for building the application? Are there any databases, glossaries, thesauri, etc. which describe the relationships between the domain objects?

2.2 What external knowledge sources can be consulted?

2.3 Which domain experts are available to support the knowledge engineering process in general? Who will verify the similarity model built by the knowledge engineers?

Business Model

Even before starting to design the application in detail, it has to be worked out what business processes are required for successfully implementing and running the system. If required, new processes have to be defined, such as for the maintenance of the system. In particular, the questions include:

3.1 Who will be running the system? What are hardware and software requirements? Will there be security issues involved?

3.2 Who provides the original data, i.e. the *domain objects*?

3.3 Who will maintain the system? More precisely, who is allowed to perform changes on the case data and who may change the similarity model?

3.4 Who will be using the system? What kind of equipment can be assumed? Will the system be free of charge?

3.5 Will there be a feedback from users? How should this be evaluated?

Questions 3.1 to 3.4 can also be summarized as: Who will be the involved actors?

Application Scenario

Even after having clarified who the actors of the application will be, a more detailed description of the group of users is required. This includes questions, such as

4.1 Who will be the users? Will they all be members of the same organization, a selected group of users outside the organization, or is the application open to anybody?

4.2 As far as knowledge about the application is required, can a homogeneous group of users be assumed? Will they use the same terms for querying the system? If asked, would they agree about the answers received for a query?

4.3 Are there limitations for certain groups of users, for example because of security reasons?

In particular, the answers to questions 4.1 and 4.2 are crucial as they indicate whether or not it seems promising to build a single application and what should be taken into account when doing so.

Remember that similarity in Case-Based Reasoning is a kind of *estimated utility*. If, however, there are groups of users which would — because of different expertise, background, motivation, or whatever reason — consider completely different cases as useful, then building a single application with a single similarity model is doomed to failure right from the beginning. Also, these questions should clarify how reliable information provided by the user is and what the typical *vocabulary* will be.

Compared to question 4.1, 3.4 addresses more technical issues, such as accessibility or fees for the usage of the system.

Tables 8.1 on page 107 and 9.1 on page 135 summarize the answers to the above questions for the VIRTUAL TRAVEL AGENCY and the SIMATIC KNOWLEDGE MANAGER, respectively.

7.3.2 System Specification

After the domain has been analyzed carefully, the next step in building a CRN-based information system is to define the structures that are required for constructing a Case Retrieval Net. This is the task of knowledge engineers but the results have to be verified by the domain experts.

Case, Case Base, Query

Cases should, of course, correspond to the *domain objects*. Therefore, the first question that has to be addressed is what is an appropriate form of representation: Is it sufficient to simply implement cases as feature vectors of fixed length or are more flexible case representations required?

Also, it has to be defined which parts of the *domain objects* will be represented in cases. As these cases will later be used to construct a Case Retrieval Net, only those components have to be included in the case representation which really may contribute to the process of case retrieval. In any case, the definition of a proper case format is closely related to the specification of *information entities* (see below).

Then, processes have to be defined for obtaining the *case base* from the data available in the domain. In particular, when this data is changed frequently, an automatic conversion of *domain objects* to cases is required.

A related question is whether or not a *case base* is really constructed based on the set of *domain objects*. In some domains, it may be more reasonable to consider cases as *views* on the domain objects in the sense of database technology: Whenever, information about cases is required, this *view* is used to directly extract the information from the *domain objects*. This may require additional effort while building a Case Retrieval Net but has the advantage of not having a second set of data structures *besides* the original data. Hence, consistency between the *domain objects* and the *case base* is easier to preserve. Of course, this would require permanent access to the *domain objects*.

Finally, the nature of a *query* has to be specified, i.e. what may be entered by a user of the system when searching the case data. In many situations, the format of a query will directly correspond to the format of cases, but there may be circumstances where the query is represented differently or does only contain some parts present in the cases.

Also, it has to be specified whether or not the user may use personal preferences in the sense of *dynamic weighting* as discussed in Section 5.5.1.

> **Example 7.6** In the VIRTUAL TRAVEL AGENCY, a case base is constructed automatically by parsing the data provided by tour operators. Cases are defined to be feature vectors for a set of predefined attributes, and queries are represented the same way. User-defined preferences are currently not supported.

Information Entities

Information entities are an essential building block for constructing a Case Retrieval Net. Remember that an IE represents a single element of a case, i.e. a knowledge item on the lowest level of granularity of knowledge representation in the given domain.

Consequently, for most applications a first approximation of the set of all *information entities* can be obtained by investigating the *domain objects* and analyzing which knowledge items occur, i.e. which feature values, keywords, etc. In terms of the knowledge containers of Section 2.2.5 this corresponds to identifying the vocabulary used for describing cases.

In addition to all the IEs present in the case base, however, it may also be required to *anticipate* further IEs which are not present in a given case base but may occur in a query. If the set of all IEs, including the *anticipated* ones, remains finite, these may be represented explicitly in the CRN. Otherwise, the concept of *computational nodes* as discussed in Section 5.5.2 has to be used.

> **Example 7.7** Queries in the VIRTUAL TRAVEL AGENCY may, for instance, refer to some destinations which are not present among the current set of offers. Consequently, the set of all IEs in that application has been defined according to all feature values that may *potentially* occur in cases or queries. As this is a finite number, computational nodes are not required.

Similarity Model

The *similarity model* should describe the similarity relationships between all IEs as defined in the previous section. When a composite similarity measure should be applied, then it suffices to specify the relevant attributes and to define the relationships between IEs representing values of the same attribute.

In general, all the knowledge sources as identified in the *domain model* can be utilized to build the *similarity model*. Depending on how reliable this model is and how often it will change, it may be required to explicitly represent this model, for example by means of ontologies, and to provide some tools for modifying it.

Relevances

The last component required for building a Case Retrieval Net are the *relevances*, i.e. statements about how important each single IE has been for a particular case. In fact, we have to distinguish between a static (*default*) relevance and a dynamic relevance. The former is encoded by means of the relevance arcs in a CRN whereas the latter may be expressed as part of a query, e.g. by weights associated to certain features.

Again, when using a composite similarity measure, specifying relevances is simpler than in the general case because only those IEs that represent the feature values of a case matter for assessing the similarity of that cases to a query. Hence, the relevances of other IEs to that case can be safely neglected. In general, the *domain model* must provide information which can be utilized for defining *relevances*.

7.3.3 System Implementation

The next step in building a CRN-based information system is, of course, to implement all the components discussed in Section 7.2 based on the specification derived during *domain analysis* and *system specification*.

We will not go into further details about these components here as they have been presented in depth in the beginning of this chapter. It should be clear that in particular the *business model* and the *application scenario* will strongly influence implementation details, for example with respect to hard- and software requirements or matters of security.

Detailed descriptions of implementation issues can be found in Sections 8.2.6 and 9.3.6 when application projects are described in depth.

7.3.4 System Maintenance

The last step which has to be dealt with is often ignored at least in academic prototypical systems. It is concerned with maintaining the system and keeping it up-to-date.

In general, any of the knowledge containers listed in Section 2.2.5 may change during the life cycle of a Case-Based Reasoning system. We will, however, restrict ourselves here to changes in the case data, the similarity model, and the vocabulary used because adaptation knowledge does not go into the Retrieve phase and, hence, is not encoded in a Case Retrieval Net.

Case Base Maintenance

Obviously, the case base will be the component of a CBR system which changes most frequently. In some situations, the case base may only be extended, i.e. new cases will be added to the system, while in other situations the case data may be exchanged completely.

As discussed in Section 7.3.2, it is desirable to automatically convert the *domain objects* to case representations in order to reflect changes in the underlying data. If this can be implemented, the construction of a new Case Retrieval Net based on the new case data suffices to keep the system up-to-date.

If there have been only minor changes to the case data, such as single cases being added, then the CRN can even be updated *incrementally*, i.e. by inserting single nodes for the new cases as discussed in Section 5.3.3. While this is primarily an aspect of system implementation, it should also be discussed from a maintenance point of view in particular when building the CRN from scratch is time-consuming.

Similarity Maintenance

In addition to the case data, the similarity model may need modifications while the system is in use. Once changes to this model have been done, a new Case Retrieval Net can be constructed or, if there have been only minor variations, the existing net may be modified incrementally again.

Vocabulary Maintenance

Last but not least, the vocabulary used to describe cases may change. Whether or not this can be captured in an automatic way, heavily depends on the application. More precisely, it depends on whether the influence of this change on case retrieval can be automatically determined. This touches questions such as:

- Does the updated vocabulary need a change in case representation, as for example when adding a new feature to a vector-based representation?

- How does the updated vocabulary relate to the representation of the *domain objects*? Does the change affect the conversion of *domain objects* to cases?
- If new vocabulary elements have been added, can the similarity relationships to the existing vocabulary be automatically derived? What about the relevances?
- What happens if vocabulary elements are being removed which have been present in case representations?

Maintenance Processes

Apart from the technical implementation details, however, *business processes* have to be defined which specify when an update of the above components is performed and who is allowed to modify the contents of these knowledge containers (cf. question 3.3 in Section 7.3.1).

Also, an important question is whether the system has to be accessible while an update is performed or whether it may be switched off during this process. This is particularly important if the update process is time-consuming due to the amount of data that has to be processed.

Chapter 8

CBR for Product Selection and Evaluation

Knowledge systems are an essential weapon in the global drive for faster product development and service delivery, higher quality, and decreased costs.
F. Hayes-Roth and N. Jacobstein, Comm. of the ACM, Vol. 37, March 1994

In this chapter, we will in full detail present a project in which the CRN model plays a crucial role: The VIRTUAL TRAVEL AGENCY *is a WWW-based application providing access to huge data sets containing tour packages by a major German tour provider. In that domain, the* VIRTUAL TRAVEL AGENCY *is one of the most successful projects in Germany.*

Following the detailed description of the VIRTUAL TRAVEL AGENCY*, we will present some projects with a related background more briefly.*

8.1 Basic Ideas of E-Commerce

Due to the wide spread availability of the World Wide Web (WWW), a strong tendency can be observed to utilize this medium for commercial purposes, i.e. for providing product information, selling products, and offering customer support.

In principle, there are three different but related sub-areas of what is traditionally called *Electronic Commerce* (see also Wilke 1997; Wilke, Lenz, and Wess 1998):

Pre-Sales: A potential customer is provided with all required information about appropriate products or services. The most obvious way of supporting this stage of Electronic Commerce are *electronic product catalogues* as they are widely used throughout the WWW.

Sales: Customer and sales agent negotiate about products and services, including prices, special discounts, delivery etc. (Beam and Segev 1996; Beam and Segev 1997). Usually this is a too complex process to be handled by traditional IT systems.

After-Sales: When customers face problems with products or services they have bought before, they might need additional support for actually using the product. The simplest way of providing this is via web sites containing lists of *Frequently Asked Questions* or via special hotlines.

In this chapter, we will present applications that deal primarily with the first task, namely the development of technologies that assist the customer in finding an appropriate product, i.e. a product that best suits his/her needs. As we shall see, Case-Based Reasoning techniques can play a key role in such systems although a number of other factors have to be taken into account, too.

Chapter 9 will address the *After-Sales* issue, in particular the utilization of CBR techniques for building an intelligent customer support system.

8.1.1 Requirements for E-Commerce Applications

In general, electronic marketplaces require special techniques in order to provide customers a real benefit from using these rather than buying products in a traditional shop. The problems to be addressed include

a) availability of the system;

b) issues of security and authenticity;

c) matters of presentation, product selection, and customer support during all three phases of Electronic Commerce.

For **(a)**, the general availability and accessibility of an E-Commerce system, the World Wide Web plays, of course, a key role. Using this medium, customers all over the world can be reached without the company really being present in that region.

Concerning **(b)**, security and authenticity, a whole range of techniques have been developed which (when applied correctly in its entirety) could help to overcome the retentions that still exist today on the customers' side.

Finally, for product selection **(c)**, techniques originating in Case-Based Reasoning appear to be promising candidates, in particular for the development of intelligent systems assisting the customer in finding appropriate products according to given requirements.

8.1.2 Intelligent Support for E-Commerce Applications

Today, a growing number of fielded E-Commerce applications can be found on the WWW. Examples include

- *Dell*'s highly successful direct marketing of computer hardware[1];
- virtual book shops, such as *Amazon*[2] or *telebuch*[3];
- CD-ROM shops, such as CD4YOU[4] or *teleCD*[5];
- traveling applications, both for business travelers, as for example hotel reservation services[6], and private holiday tours, such as the VIRTUAL TRAVEL AGENCY which will be presented in detail below.

As far as product selection is concerned, most of these applications still rely on standard database technology. This has the advantage of being highly robust and efficient but an intelligent support of customers can hardly be implemented with this technology. In fact, such electronic shops can be considered as *silent E-stores* in which customers are left alone with their needs.

[1] www.dell.com
[2] www.amazon.com
[3] www.telebuch.de
[4] www.cd4you.de
[5] www.telecd.de
[6] www.hrs.de

What do we mean by this? As long as there is only a fairly limited number of products in an E-Commerce application, it is sufficient to list these all on a web page and possibly to include some kind of structuring according to product groups etc. If, however, there is a large number of products, then some kind of query interface is required by means of which a customer can enter his/her requirements and search the product database for an appropriate offer. Given this, three principle situations might occur:

- Firstly, the customer might know precisely what s/he is looking for and how this specific product is referred to in the product database. Also, this product is available. Then, the customer will be lucky as s/he has found directly what s/he was looking for.
- Secondly, the customer might still have a precise desire but either s/he does not know what the name of the product is in the database or the product may be currently out of stock.
- Thirdly, often situations occur in which customers do *not* know precisely what they are looking for. Rather, they have some kind of need they want to fulfill and a more or less vague idea of what the solution might look like.

Standard database technology has in particular problems in dealing with the last two situations. Obviously, some kind of *knowledge* is required in order to suggest alternative products and to infer from the customer's needs which products might be appropriate. An example for the deficiencies of standard databases is given in Section 8.2.2.

This knowledge has to be represented and dealt with in addition to the pure product database. Hence, knowledge-based techniques are required; and, as is shown in this chapter, CBR is among the key techniques.

We do, however, only consider a very limited aspect of E-Commerce here which is based on the ideas of information completion as described in Chapter 3. For example, we do *not* deal with negotiation processes, configuration of appropriate products, or intelligent assistants guiding the customer through the selection and sales processes.

8.2 The VIRTUAL TRAVEL AGENCY

8.2.1 Description of the Domain

As mentioned already in Section 1.1.1, CABATA (Lenz 1994b) has been a prototypical system illustrating the ideas of CBR by implementing a decision support system in the domain of tourism. After CABATA had been in discussion for some years, Humboldt University started a cooperation with *check out*, a Berlin tourist office chain which aimed at building a running application used by the tourism industry.

From a research perspective, the objective of this cooperation was to get access to real data of major German travel agents and to apply the ideas developed within CABATA for building interactive World Wide Web-based kiosks. From the practitioners side, the goal of *check out* was to make use of WWW facilities for their daily business, i.e., to allow potential customers to browse their offers and perform online booking.

Handling of Last-Minute Offers

A major problem in the daily business of travel agents is the handling of *Last Minute* offers, i.e. tour packages that become available only some days prior to the departure date and that can be purchased for a reduced price. The market for *Last Minute* offers is steadily growing and there are specific properties related to these products:

Property 1: Travel agents struggle with the *update* problem: Up to 6,000 of new packages are offered daily just by one of the major German tour providers. Traditionally, tour providers sent several dozen sheets of paper to their travel agents every day in order to inform them about available special offers.

Property 2: The above described way of informing agents via stacks of papers implies an *availability* problem: Often, the amount of places that can be booked on a specific offer is highly limited. Hence, when customers decide for one of the offers, there is a high risk that this one is no longer available. This happens because there is no feedback about offers that have been brought to the market in recent days, i.e. it is not clear whether these are still available or not.

Property 3: Thirdly, *Last minute* offers (as provided by the tour providers) are tour packages: This means that the customer may accept an offer only as it is – there are no variations of it (except if it is stored as a separate offer). Consequently, there is no negotiation during the sales process.

Property 4 Finally, although there may be a huge number of offers, it is unlikely that the desires of a customer can be fulfilled all at once. Rather, it is often the case that alternative departure dates, neighboring airports, or even other destinations need to be suggested. In contrast to people having planned and prepared their holiday carefully, customers looking for *Last Minute* vacations expect such variations.

It is obvious that the vast amount of information that has to be handled for the *Last Minute* market can only be dealt with using appropriate tools. In particular, both the *update* and the *availability* problem can only be solved by distributing and updating the data electronically.

The third property implies that there is no need for extensive counseling by the travel agents. Instead, customers may be provided with all the information that is required to decide for a particular offer. Even tickets may be handed over to the customer at the airport. Hence, there is no need for a personal contact between the customer and the travel agent and, thus, this domain is well-suited for building a WWW-based kiosk.

However, property 4 indicates that a standard database technology will not suffice — just as discussed in Section 8.1.2.

Providing Additional Information

As mentioned above, Property 3 makes it possible for the customer to use the services provided in a self-service manner. However, this requires that, apart from the pure data about tour packages, additional sources of information as well as other services are made available via the same system.

This includes

- information about destinations, countries, and regions;
- information about climate, health care, and visa regulations;
- services which are often used in addition to booked packages, such as insurance or car rentals.

In fact, the objective of the VIRTUAL TRAVEL AGENCY was to consider all services provided by traditional travel agents and to provide as many as possible via the World Wide Web.

8.2.2 Motivation for a CBR Approach

As already discussed in Section 8.1.2, simple database approaches would not be sufficient for implementing this type of application as they cannot provide *intelligent* sales support. In particular, the following features are essential:

- Some customers enter very detailed descriptions of their intended tour packages while others only have a rough idea. Consequently, the system has to be able to deal with vague as well as highly specific queries.
- Due to Property 4, the system has to be able to suggest appropriate alternatives if no offer completely fits the customers requirements.
- The system has to present the available offers to the customer in a reasonable manner. In particular, it should definitely avoid situations in which no offer is made to the customer (*no solution* situation) as well as those where the customer is left alone with pages and pages of possible offers (*1,000 solutions* situation).

Case-Based Reasoning, in general, is a technology that satisfies at least the last two criteria:

- Alternatives are suggested by considering *similar* offers where the similarity measure takes into account information about departure dates, geographic locations of departure airports and destinations, climatic conditions etc.
- The *no solution* situation is avoided by the previous point: consideration of alternatives. The *1,000 solutions* situation, on the other hand, is avoided by establishing preference orderings and presenting the best suited offers first. Hence, the result of a retrieval process is not a *set* of all applicable offers (as in databases) but an *ranking* of the most suitable offers.

Fulfillment of the first of the above criteria very much depends on the particular technique used for implementing the system (see Section 8.2.3).

According to these requirements, the basic ideas of the CaBaTA prototype are directly applicable, i.e. there is no need to implement, for example, an additional planning component for configuring an appropriate offer. Compared to CaBaTA, however, one has to deal with efficiency problems (due to the huge number of offers) and one has to provide solutions for the problems specific to the implementation of an online WWW system, such as online booking, regular updates, etc.

8.2.3 Advantages of CRNs

As should be clear from the description so far, a CBR system has to deal with three crucial problems in the Virtual Travel Agency:

Flexible Retrieval: Some customers enter very specific requests, others only express vague intentions. Hence, a CBR system needs a flexible retrieval method that can cope with both situations.

Efficient Retrieval: Customers are not willing to wait long for the result of a search request. Even the delay caused by the WWW often causes severe problems. Thus, the retrieval method has to be highly efficient due to the large amount of data that has to be managed.

Easy Updates: Since regular updates of the data are required, building the internal structures of the CBR system has to be manageable in reasonable time and possibly while the system is still running.

As discussed in Section 5.3, Case Retrieval Nets satisfy all three criteria.

On the other hand, completeness and correctness, another important property of BCRNs, are of minor importance here because these formal properties are always considered in terms of the similarity measure given on the representational level SIM_{Rep}. In that particular application, however, there is obviously a highly subjective factor in assessing the similarity of regions, destinations, and dates: While a certain customer might consider the two Caribbean islands `Barbados` and `Guadeloupe` as highly similar because of their geographic location, some other customer might disagree because of the different languages spoken. Similarly, while *Last Minute* customers are, in general, flexible with respect to the departure dates, there may be hard constraints for a particular customer which does not permit booking an offer which leaves just a single day later. Consequently, the similarity measure in SIM_{Rep} will already be imprecise as it is not able to capture the subjective assessments existing on the application level SIM_{App}.

8.2.4 Domain Analysis

Given the data and the desired functionality of the the *Last Minute* application, the description of the domain according to the criteria listed in Section 7.3.1 is straightforward:

Domain Objects

The *domain objects* are, of course, given in the data sets provided by the tour operators, that is each object corresponds to a single tour package. Furthermore, we assume that the data sets are given in a predefined format which allows for an automatic parsing and conversion to a table-like format.

Domain Model

The *domain model* should list all the features that could potentially be used for describing a tour package. In addition, the *domain model* has to describe similarity relationships for all possible feature values. Both, the features and their values can be directly derived from the provided *domain objects*.

Currently, the following features are present in the data:

- the `Departure Airport`,
- the `Destination` of the desired holiday trip,
- the `Departure Date`,
- the `Duration` of the trip,
- the `Type of Accommodation`
- the `Type of Catering`,
- the number of participating `Persons`,
- and, of course, the `Price`.

Building the domain model is either straightforward (as, for example, for the features `Departure Date` and `Duration`), can be based on general geographic information (as for the `Departure Airport` feature), or has to be done by domain experts (as in the case of the the `Destination` feature).

Given this, an important property of this application is that the entire domain is finite and rather fixed, i.e. it is possible in advance to list all the potential values for all the features — except for the `Departure Date` which will depend on the actual data set provided. We

say *potential* values here because not every value is necessarily included in every data set; for example, a certain destination may be booked out.

Business Model

The *business model* for the *Last Minute* application is also quite simple. There are three different actors involved:

Tour Operators: The tour operators are only being involved in so far as they provide the data in some predefined format. They are not directly incorporated into the system. Even the booking process should be performed via the travel agents.

Travel Agents: The travel agents are responsible for

1. downloading the data sets from the tour operators[7] to allow for an automatic update of the system;
2. building and updating the *domain model* in particular for those features where expert knowledge is required;
3. processing the booking orders that are sent by customers.

Customers: Customers may use the web site to search for appropriate tour packages and use all the services provided. There is (up to now) no means of *personalizing* the system. Also, users are not allowed to perform any changes to the system.

Application Scenario

Last but not least, the *application scenario* should be clear from the description of the *business models*. We do not make any assumptions about a potential user of the system, the user is limited in his/her query by only being able to make selections in a provided HTML form.

Table 8.1 summarizes the results of *domain analysis* by briefly answering the questions listed in Section 7.3.1.

Table 8.1: Answers to the questions of *domain analysis* for the *Last Minute* application of the VIRTUAL TRAVEL AGENCY

1.1	data about tour packages as provided by tour operators
1.2	available electronically, fixed format
1.3	data will be updated twice a day in its entirety
2.1	travel agents provide a taxonomy of departure airports and destinations
2.2	no additional knowledge sources required
2.3	travel agents verify and adjust similarity model
3.1	system installed at the travel agents, no security issues involved
3.2	data provided by tour operators
3.3	case updates automatically, other changes by the travel agents
3.4	external users, access via the WWW
3.5	no feedback but booking orders
4.1	no assumptions about users
4.2	assuming homogeneous group of users due to simple application
4.3	no limitations

[7] Unfortunately, this detour is required as we have not been allowed to directly connect to the database containing all the offers at the tour operators' side.

8.2.5 System Specification

Cases, Case Bases, and Queries

A *case* is, of course, a representation of a single tour package as a vector of attribute-value pairs according to the features listed in Section 8.2.4.

A *case base* is the set of all cases in a single data set. Hence, the entire applications deals, in fact, with 3 case bases (see the description of the *Retrieval Server* in Section 8.2.6).

A *query* is represented similarly to cases except that the feature `Price` has not been included as a searchable parameter. This decision was based on the observation that all offers with otherwise similar parameters will, in general, be very similar priced in *Last Minute* data sets.

Information Entities

The *information entities* can be derived from the potential feature values of all offers (remember that we are dealing with a finite domain).

Similarity Relationships

The *similarities* of IEs can be computed based on usual distances or boolean comparisons for most features. Only for the `Departure Airport` and `Destination` features we provided a tool by means of which the domain experts could build a taxonomy from which the similarity relationships could be obtained.

As it turned out, obtaining the similarities in particular for the `Destination` required the experts' knowledge. The geographic locations could serve as a rough approximation for this. However, there are many exceptions in the sense that destinations may show quite similar characteristics despite being located in completely different geographic regions — and vice versa.

> **Example 8.1** Going on a tour through `Mexico`, for instance, shows some aspects of traveling through Central or South America. Bathing in `Mexico` is, on the other hand, similar to the `Caribbean`. Despite the common border, however, traveling to the `U.S.A.` significantly differs from usual trips to `Mexico`.

In fact, the final similarity model somehow reflected the organization of countries, regions, and cities which can also be found in the printed catalogues of tour operators.

Relevances

Finally, the *relevances* have been set based on the experts' opinions about the importance of the various features. In order to keep the user interface as simple as possible, default relevances have been defined which cannot be modified by the user.

8.2.6 System Implementation

In the current implementation, the *Last Minute* server within the VIRTUAL TRAVEL AGENCY consists of a number of modules as shown in Figure 8.1.

Retrieval Server

A *Retrieval Server* is in the very heart of the system. This server communicates via a well-defined TCP/IP protocol with the *Query and Presentation Module* (see below), performs retrieval according to the query given by a customer, and returns the best offers found.

A special feature of that server is that it does not manage just a single Case Retrieval Net but a whole set — one for each particular data set. This is because there are currently 3

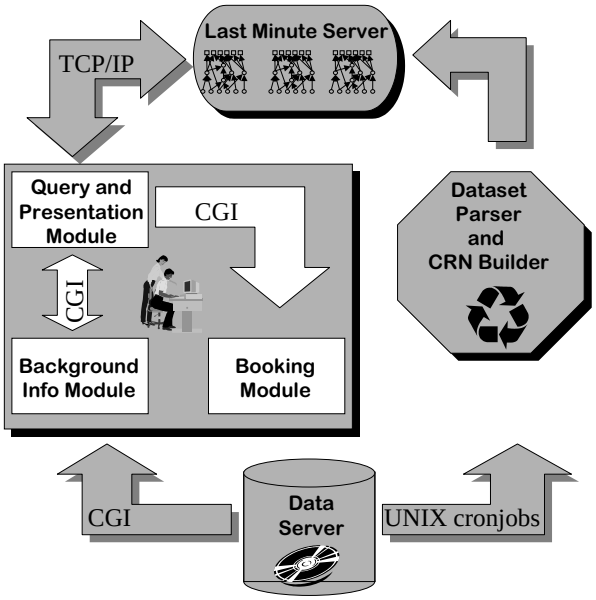


Figure 8.1: Architecture of the *Last Minute* application as part of the VIRTUAL TRAVEL AGENCY

different data sets of *Last Minute* offers, originating from different programs and even different tour providers. As it turned out during the development of the system, keeping the data for each of these programs in separate CRNs had a number of advantages:

- Each of these data sets has its own data formats and, hence, requires specific functionalities during updates.
- Apart from the original application developed for *check out*, there are also a number of customized versions (see Section 8.2.8) which, however, should have access only to some of the data sets. This is easy to implement when using separate CRNs.
- Update for each of the data sets can be performed independently of the others.

Query and Presentation Module

The *Query and Presentation Module* serves two purposes. Firstly, it provides a user interface for entering a customer's description of a desired tour package. Currently, this is a simple HTML form and the user can enter parameters, such as the preferred departure airport, the destination, the departure date and duration, the comfort class and the type of accommodation and meals desired, and so on. In the current version, these parameters are weighted internally only, a dynamic weighting as defined in Section 5.5.1 is not supported as this would cause the interface to be heavily overloaded. Here, a possible extension is to provide a *standard* and an additional *expert* interface.

Secondly, the *Query and Presentation Module* is used to display the result of the retrieval process. After having sent a query to the *Retrieval Server*, it receives the result of the retrieval process and displays the data of the suggested offers on a separate web page.

A special feature of the *Query and Presentation Module* in that application is that the query interface is built dynamically based on the given data. This means that there is no static HTML page containing, for example, all potential destinations. Rather, this page is

constructed dynamically in order to ensure that only those destinations are listed which actually are contained in the current set of offers. By doing so, we limit the problem of subjective similarity assessments as discussed in Section 8.2.3: If a destination is not in the current set of offers, it will not occur on the web interface. Hence, the user may decide about alternatives that s/he considers as acceptable.

Background Info Module

The *Background Info Module* is used to display background information for a selected tour package, such as information about the destination, the offered hotel, or the climate.

The type of information shown depends on the data set the offer originates from. For example, there are *Last Minute* programs for which the name of the hotel is intentionally not available while for other programs this information is provided.

Online Booking Module

The *Online Booking Module* implements the online booking facility of the *Last Minute* application. However, as direct access to the booking services is not (yet) provided by the tour providers, this functionality currently is limited to sending an email with the booking order to the travel agents. They then process these mails, confirm the booking, and prepare the tickets.

Update Module

The *Update Module* is required for keeping the data of the *Retrieval Server* up-to-date. During update, a new data set is parsed and transformed to a proper case representation. Then, a Case Retrieval Net is built given the new cases as well as the similarity model defined by the domain experts (i.e. the travel agents), and the *Retrieval Server* is informed about new data being available. This causes the server to discard the CRN of the corresponding data set and to load the new one.

Data Server

Based on the case representations built by the *Update Module*, the *Data Server* provides detailed information about the available tour packages, including information which is not utilized during retrieval, such as the name of the hotel (if known) or a brief description of the destination.

Further Implementation Details

In the current implementation, all of the above mentioned modules (except for the *Update Module*) are just CGI scripts as known from standard WWW programming. All components have been written in C++, the entire VIRTUAL TRAVEL AGENCY consists of 20,000 lines of code, including all additional components (see Section 8.2.8).

In order to remain flexible with respect to changes in the layout of the various pages and also to not overload the programs with too much information about the HTML layout, we implemented a strategy in which *template* HTML pages are provided by the travel agents. These are then used within the system and the data that corresponds to the current session is placed in these templates. This also made the implementation of customized releases for other partners reasonably easy.

In future versions, the *Query and Presentation Module* may be implemented by a more powerful JAVA Applet in order to improve the quality of presentation with features not supported by HTML.

8.2.7 System Maintenance

Maintenance of the *Last Minute* application has to be performed with respect to two components:

- Firstly, the available offers have to be updated regularly, i.e. at least daily. This will be performed semi-automatically based on the data provided by the tour operators (see the description of the *business model* above).
- Secondly, the similarity model may have to be changed from time to time. This will happen primarily when new destinations are added to the application; it is the task of the travel agents to make these adjustments.

8.2.8 Current State of the VIRTUAL TRAVEL AGENCY

As mentioned in Section 8.2.1, it is essential for a commercial success that data and services are provided in the VIRTUAL TRAVEL AGENCY beyond the pure *Last Minute* application. Consequently, we also implemented a number of other components which, however, are not so much of interest from an academic point of view:

- An INTERACTIVE FLIGHT KIOSK maintains a database of up to 150,000 flights, including online booking facilities.
- A RENT-A-CAR application provides access to a world-wide operating car rental agency.
- An INFORMER service uses agent-oriented technologies to maintain a database of customers' specifications for future holidays. The INFORMER *observes* the data that is provided by the tour operators, checks which of these offers might be acceptable for the customers, and sends notifications via email thus telling customers that it is worth checking the web site again.

The *Last Minute* application itself builds on the data provided by *NUR Touristik* (a major German tour provider) and *BFR* (a tour provider active in the Berlin area). During peak season, there may be up to 250,000 offers with updates required twice a day.

As mentioned before, retrieval from this amount of data can be performed on a standard workstation in about 1.3 seconds. Hence, no efficiency problems occur, in particular when taking into account that users query the system via the WWW which in most cases appears to be the performance bottleneck.

Availability and Installations

Currently, a number of customized versions of the *Last Minute* application exist:

- The main system with access to all data sets is integrated into the VIRTUAL TRAVEL AGENCY and accessible via the WWW site of our partner *check out*[8]. This web site also contains a lot of useful information and services, such as descriptions of hotels and of specific regions, overviews over climate and culture, etc. Figure 8.2 shows a snapshot of that installation.
- Another major application with access to the data provided by *NUR Touristik* has been implemented for *Holiday Land*, a group of about 200 travel agents throughout Germany[9]. In addition to the components described in Section 8.2.5, this application also contains a module for searching the nearest travel agent given a ZIP code.
- A customized version has also been built for *Berliner Flug Ring*[10] (BFR), of course only

[8] www.reiseboerse.com
[9] www.holiday-land.de
[10] BFR-Reisen.com

working on their own data.

- Finally, two versions have been installed for the TV magazine *rtv*[11] and the local newspaper *Saarbrücker Zeitung*[12].

Figure 8.2: Snapshot of the *Last Minute* application as part of the VIRTUAL TRAVEL AGENCY. As displayed, the CBR aproach allows for suggesting appropriate alternatives if a perfect matching offer is not available.

8.2.9 Evaluation

From March 1997, when the *Last Minute* application had been made available as part of the VIRTUAL TRAVEL AGENCY, to July 1998 about 500,000 search requests were processed and approximately 1,400 tour packages booked with an overall sales of more than a million DM. This commercial success requires two remarks:

- Firstly, there are similar applications on the web mainly provided by the tour operators themselves. According to the figures that have been published in the trade press, their sales is less then that of the VIRTUAL TRAVEL AGENCY. This is even more remarkable as they have direct access to all data and information required and did invest much more effort in building their WWW applications.

  The only reason we see for this is that in these applications standard databases are being used while the VIRTUAL TRAVEL AGENCY utilizes Case-Based Reasoning to provide more intelligent search facilities. This presumption is also supported by the fact that the other applications have a much worse ratio of search requests and bookings.

- Secondly, the above figures indicate principle difficulties in establishing such a project. As the travel agents only receive a certain commission depending on the overall sales and

[11] www.rtv.de/tv-reise.html
[12] www.sz-newsline.de

still have to cover their own expenses, the question arises how such a project should be financed under usual business conditions. Major parts of the VIRTUAL TRAVEL AGENCY have been developed at Humboldt University as by-products of research efforts — which only made this project possible.

We will not go into further details about these and related problems, they are discussed more broadly in Lenz (1998b, Lenz (1999).

8.3 Related Projects

Apart from the VIRTUAL TRAVEL AGENCY, a number of other projects have been performed which are related to this in so far as these projects, too, are situated in an E-Commerce scenario in the most general sense. Also, these applications are, like the VIRTUAL TRAVEL AGENCY, based mainly on structured data in the sense of cases being representable as feature vectors.

Having described the VIRTUAL TRAVEL AGENCY in detail, it suffices to only sketch the related projects and to omit the details. The main motivation for listing these applications here is to give an idea of further potential application areas and, in the case of CBR-SELLS, to show how an application-independent system might look like.

8.3.1 Real Estate Assessment

The real estate business is another very promising area for applying Case-Based Reasoning techniques in general and Case Retrieval Nets in particular. The objectives for building a decision support system for this area may be twofold:

- Firstly, a goal may be to provide some kind of electronic product catalogue to the end-consumer similarly to the VIRTUAL TRAVEL AGENCY described above. Users may list desired features of an estate they are looking for and are presented with the best matching offers found in some database of a real estate agent.

- Secondly, experts working in the field (including real estate agents) may be supported, for example, when assessing the value of an estate that is for sale.

For the latter point, i.e. for estimating the value of some flat, house, or property, two different methods are being used by domain experts today (Ladewig 1994):

Estimated Income Value: On the one hand, the value may be determined based on the estimated profits, for example, from renting out flats or offices on the estate. This method is mainly applied to buildings with offices, apartment houses etc.

Comparative Value: On the other hand, the value may be estimated by comparing the given estate with similar estates on the market and applying some adaptations according to the differences found. This technique is most often applied to single family homes and estates mainly used for private purposes.

Obviously, the latter approach is closely related to Case-Based Reasoning and, for example, Gonzalez and Laureano-Ortiz (1992) already suggested a CBR approach to real estate appraisal.

In the following we will briefly present the basic ideas of the FIB[13] system that has been developed in cooperation with staff of a major German bank. A more detailed description has been given in Lenz and Ladewig (1996).

[13] Fallbasierte Immobilien-Bewertung — case-based real estate assessment

Description of the Domain

The task of the domain expert during real estate assessment is to determine, as precisely as possible, the current value of an estate. For this, many factors have to be taken into account some of which may be known to the expert (such as the size and equipment of a flat) while others are hard to determine (such as the potentials for future increase in value).

The development of decision support systems for that area is challenging due to a number of factors (Ladewig 1995):

- There is a huge amount of information that has to be taken into account which is hardly manageable without IT support.
- In general, an algorithmic solution will not be possible but heuristics are required to estimate the influence of various factors.
- Expert knowledge and experiences play a crucial role in particular when applying the *Comparative Value* procedure.
- Often, decisions have to be made or at least prepared by staff who only rarely have to deal with such tasks.
- Despite different estates being evaluated by different members of staff, it is desirable to establish some common standards, for example within a single broker company. Clearly, this addresses issues of an *organizational memory* as discussed in Section 1.1.2.

Motivation for a CBR Approach

Ladewig (1994) listed a number of techniques that might be used for building a knowledge-based decision support system for the tasks described above. As it turned out, Case-Based Reasoning is particularly useful because of several reasons:

- Every single estate directly corresponds to a single case. In fact, domain experts use terms which already suggest a case–oriented approach.
- Similarity of estates directly corresponds to similarity of cases. Furthermore, there are a number of factors common to classes of estates (such as the `size` of a flat) which can be used to base similarity assessment on.
- A CBR system can be extended at any time by inserting new cases. Hence, by simply adding estates and their (confirmed) values, the Retain phase of the CBR cycle (see Section 2.2) can be implemented.
- CBR can be considered as a direct *implementation* of the *Comparative Value* procedure. Whether adaptation as part of the Reuse phase is performed within the system, depends on the amount of adaptation knowledge integrated. In general, the diversity of factors that influence this process will not permit a completely automated problem solving but require the integration of the domain expert.

A problem, however, that would need further explorations is that the value of estates not only depends on factors that are directly associated to that estate but also on the situation on the market and on time.

Advantages of CRNs

The main reason for using CRNs as the underlying model for case retrieval has been their flexibility as discussed in Section 5.3.3.

In particular, the idea of considering cases as arbitrary sets of information entities is highly useful because estates can, in general, not be described by a fixed set of features (as was the case for the *Last Minute* application). Rather, each property, house, or flat may have its own characteristics apart from features that are shared at least by a certain class of objects (such as the `size` of a flat or the `type of heating`).

Another advantage is that Case Retrieval Nets implement an information completion process as discussed in Chapter 3. Consequently, not only can the value of an estate be estimated based on values of similar estates, but the reasons for potential differences are listed, too.

> **Example 8.2** When evaluating a single family home, the expert may observe that there are two groups of houses: cheaper ones and more expensive ones. When considering the retrieved cases in detail, it may become evident that the reason for this is that the latter houses all have an outdoor pool which the cheaper ones do not have. Consequently, the expert will check whether the current estate has this feature, extend the case description accordingly, and obtain a more precise estimation of the value.

Given this, there is no reason why such a CBR system should be specific for the purpose of real estate assessment. Rather, if it implements the search for similar estates, then it may as well be used for building a system supporting a potential buyer when searching for an appropriate house or flat.

Domain Analysis

To test the applicability of CBR techniques for real estate assessment, we decided to first build the FIB prototype working on data provided by the *Bayerische Hypotheken- und Wechselbank*, München. The data set contained approximately 3,000 descriptions of apartments in Berlin collected over a period of 3 months.

The *domain objects* did directly correspond to these data records, that is each entry describing a single apartment is considered a single domain object.

The *domain model* has been derived on the basis of the data provided by the bank. In particular, the following features have been used to describe apartments:

- the `Number of Rooms`,
- the `Size` in square meters,
- the `District` within Berlin,
- the precise `Address` of the apartment,
- the `Floor` the apartment is on,
- whether a `Lift` is available,
- whether it is a `New Building` or not,
- and, of course, the `Price`.

For most of these features, building the domain model is straightforward as these are either boolean or numeric features. The precise `Address` has not been used for similarity assessment. For the `District` feature we determined the similarity of the various values based on

- the geographic location,

- the distance to the city center,
- the social population structure,
- the means of public transport,
- the shopping facilities,
- the cultural and leisure time facilities,
- the typical type of buildings.

In fact, this type of similarity assessment corresponds to the model of microfeatures as presented in Section 6.2. However, for this application microfeatures have not been integrated directly in the Case Retrieval Net model. Rather, they have been used off-line to determine the degree of similarity for the various values of the `District` feature.

As FIB was designed as a prototype, the *business model* as well as the *application scenario* did not play a major role. The intended group of users were the employees of the bank working in the area of real estate assessment. The prototype has been designed as a *static* application, that is no update or maintenance issues have been addressed.

System Specification

As should be clear from the analysis of the domain, each *case* represents, of course, a single data record, i.e. a single apartment.

The *case base* is the set of all those cases, we did not determine the source of the cases or distinguish these with respect to the time when the data had been collected.

A *query* directly corresponds to a case, i.e. it has exactly the same structure.

The set of all *information entities* has been obtained by analyzing the data and determining all potential feature values. As was the case for the *Last Minute* application, the domain is finite for a given case base.

The *similarities* of IEs have been determined as described above. Basically, only the `District` feature caused some degree of knowledge engineering while building the similarity model was straightforward for all other features.

Finally, the *relevances* have been defined according to the opinion of a domain expert about the importance of all the features.

System Implementation

FIB had been implemented as a stand-alone WINDOWS program. Consequently, it contained all the modules which are required for a CRN–based decision support system according to Section 7.2:

- The *Retrieval Server* had been realized as a single Case Retrieval Net which has been a module of the entire program.
- The *GUI* was a standard WINDOWS interface including a form for entering a query, a table for browsing the retrieval results, and a screen for viewing detailed information of a case.
- An external *data server* had not been used. Rather, all information about the data sets was managed by the FIB program itself.

System Maintenance

Due to FIB being a prototype, issues of maintenance have not been addressed.

Current State

The FIB system had been prototypically implemented in 1996. A carefully directed evaluation by the domain experts never took place due to a number of reasons. Up to now, FIB has been used mainly for demonstration purposes, a demo on the WWW based on the same data is provided today by *TecInno*[14].

Apart from these demo programs, no further activities emerged up to now. But as more and more real estate agents start to make their offers available on the World Wide Web[15], it might be worth another trial.

8.3.2 CBR-SELLS

Today, most so-called electronic shops in the Internet are implemented with standard database technology. A crucial drawback of this is that potential customers cannot be supported when searching for products that suit their needs. In particular, there are two major requirements that must hold in order for such an approach to be promising:

- Firstly, users must be able to precisely and unambiguously describe what they are looking for.
- Secondly, there must be a perfect match from the users' descriptions to the available products.

If either of these requirements is violated, then the users will be shown hundreds of offers that are claimed to be useful, but in fact only overwhelm users. Or there will be no offer at all, leaving frustrated customers behind. There is no support in the sense of having a sales agent telling the customers what is wrong about their requests or what alternatives there are (Wilke, Lenz, and Wess 1998).

The objective of CBR-SELLS is to provide a generic tool, or library, allowing makers of electronic shops to benefit from Case-Based Reasoning techniques in general and the advantages of Case Retrieval Nets in particular.

Description of the Domain

Both systems described in the first part of this chapter have been developed with a specific application in mind: Although the Case Retrieval Net model has been used for retrieval, the systems differ significantly with respect to case representation, similarity model, and integration issues.

After having investigated these and a number of other potential application areas, it appeared that the scenarios for applications in electronic commerce are highly similar:

- Often, an attribute–value based representation is sufficient, such as encoding tour packages by means of features as `Destination`, `Comfort`, or `Price`.
- The product data which should be used for building a case base is often available in standard (most often relational) databases.
- In most cases, a composite similarity measure is sufficient and the similarity model is straightforward in the sense that domain experts (i.e. salesmen) are well able to describe which features are related to each other and what influence these features have on customers' decisions.

[14] www.tecinno.com
[15] See www.rdm.de, for example.

- Often the most important task in these applications is finding appropriate products. On the other hand, adaptation or even integrated learning, which both require highly domain-specific approaches in most cases, are only of secondary importance.

Consequently, a generic tool for enhancing such applications with Case-Based Reasoning techniques might be desirable. More precisely, a company developing electronic shops should be provided with methods and tools for describing their specific similarity model and for transforming the product data into case bases. Then, a case memory can be constructed which can be shipped together with the product descriptions intended for usage by the customers. The latter may access the system via customized user interfaces and enter their requests. However, this request will be answered by a case-based retrieval over the product data rather than by a standard database transaction. Apart from more appropriate answers by the system, such as suggestions for reasonable alternatives, this paradigm change will be invisible to the user.

Motivation for a CBR Approach

The main benefits from using CBR technology should be clear from the discussion so far. For the developer of an E-commerce application, the main advantage is that s/he will be able to integrate case-based retrieval methods with only a limited overhead on implementation effort. Thus, knowledge of salesmen can easily be incorporated into the system. For the user, of course, the benefit is in getting more intelligent answers by such a system — such as avoiding 1,000 solutions or no solution at all.

Advantages of CRNs

A particular advantage of CRNs is that, for a given data set, such a memory model can be constructed and shipped independently of the real product data. Thus, on the user's side it is not required to install a complete CBR system which then has to be maintained. Rather, a simple library suffices which performs retrieval based on the case memory and then uses an external database to access the details of the retrieved products.

In particular, the latter database not necessarily has to be the same as the one the CRN has been built for. For example, one might imagine building a product catalogue based on an internal database in which all details about wholesale prices and other confidential information is included. The customers, however, will only be provided with the data required for satisfying their information needs. Of course, there must be a well-defined relationship between the two databases ensuring that the correct products can also be found in the database shipped to users. Usually, a *primary key* referencing each product record will suffice for this purpose.

Domain Analysis

As discussed already, CBR-Sells has not been designed for a specific domain but for electronic commerce applications, in general, which share the above listed characteristics with respect to case representation, similarity model, and focus on retrieval tasks. To facilitate this goal, the CBR-Sells library currently supports the following types of attributes as *built-in* types:

Numeric data type used to represent integer and floating point numbers. For this data type, a built-in similarity measure is used. More precisely, there are three sub-types implementing similarity measures based on inverse distances, on a preference of lower values, and on a preference of higher values, respectively.

Symbolic data type used for representing non-numeric data. For this type, similarity of values will be determined based on a similarity matrix which has to be specified by the developer of the application. This should be general enough to cover all situations, including taxonomies and (partial) orderings.

String data type used to represent, for example, product names for which a matching based on regular expressions might be desirable, such as when requesting `DeskJet5*C` to search for *Hewlett Packard*'s color ink-jet printers of the 500 series.

Date data type used to encode dates and time stamps within the product data.

Taken together, all these types of attributes provide sufficient flexibility for describing most E-commerce applications. In particular, these would allow to re-design the VIRTUAL TRAVEL AGENCY such that this could make use of the CBR-SELLS library.

System Specification

As CBR-SELLS is designed as a general-purpose library, the developers of electronic shops have to specify where the information required can be obtained from. In particular, they have to provide product data to build a *case base* and a case memory from. Again, a *case base* is not really constructed but as discussed in Section 7.1 it can be considered as a specific *view* on the existing database(s).

The set of *information entities* will be constructed based on this product data by extracting all the various feature values occurring in the database.

The *similarities* are either computed by built-in functions or can be specified by the developer — based on the chosen feature types.

Finally, the *relevances* can either be specified by the developer or can be contained in the queries expressed by the actual users by specifying appropriate weights.

A crucial difference to the applications described so far is that the library should be domain-independent. Hence, all parts which require customized solutions have to be developed by the company building the electronic shop, such as specific interfaces allowing users to describe their requests and presenting the product information in an appropriate manner.

System Implementation

CBR-SELLS consists of two separated modules (cf. Figure 8.3):

- Firstly, a BUILDER has to be used for developing an electronic shop. More precisely, the system has to be informed about the application domain, i.e. what products there are, how they are described, and how they are related. Based on this, a Case Retrieval Net is constructed by the module of the system which may then be used by the FINDER module for searching product databases.

- Secondly, the FINDER module is used to load the CRN constructed by the BUILDER and to answer queries by performing a case-based search over the specified data set.

The BUILDER should be used whenever a product catalogue has to be shipped, for example on a CD-ROM. Building the index has to be done once for every data set. The FINDER is then really used by the customers who search for specific products. For both phases, a set of well-defined functions are provided by the library.

Figure 8.3 shows the principle architecture of the system and the two phases of using it. For the first phase, a *Builder GUI* is provided that allows developers to construct their similarity model and case memory in a convenient way. This GUI has been developed by *TecInno*[16].

For the second phase, the *Retrieval GUI* has to be implemented by the company developing the electronic shop because only they know how to present the products in the most appropriate way and where to obtain additional information, such as pictures, sounds, and videos.

[16] www.tecinno.com

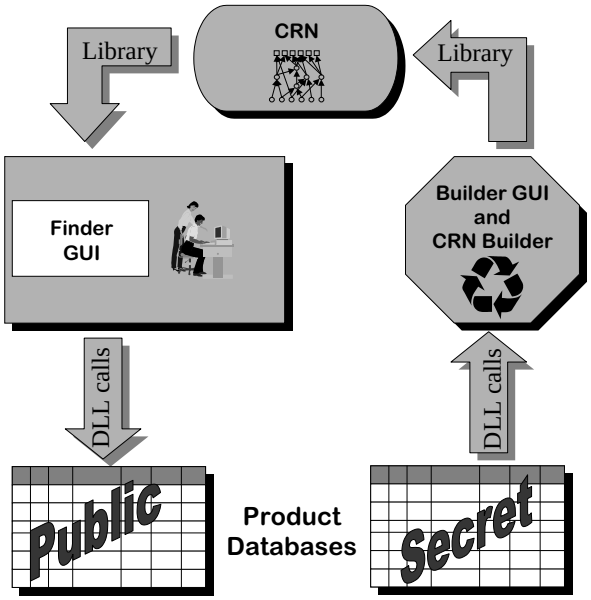


Figure 8.3: Architecture of CBR-SELLS

System Maintenance

According to the requirements of most electronic shops, maintenance is only required on the BUILDER's side. Once a product catalogue is shipped, this cannot be changed. If significant changes to the data occur, then a CRN has to be constructed by using the BUILDER module again. Of course, parts of the similarity model etc. can be stored by the BUILDER and do not have to be reconstructed from scratch every time.

Current State

Currently, CBR-SELLS is available as a 32-bit DLL for Windows-95 and Windows-NT. On request, libraries for other operating systems, first of all UNIX, could also be provided. Recently, it has been shipped for a CD–based product catalogue of *Analog Device*[17].

[17] www.analog.com

Chapter 9

CBR for Knowledge Management

All the evidence suggests that for end-user searching, the indexing language should be natural language, rather than controlled language oriented.
Lewis and K. Sparck Jones, Communications of the ACM, 1996

In this chapter, we will present a fairly new direction of CBR research, namely Textual CBR which addresses the handling of textual information rather than structured data. Using a recently performed project, the SIMATIC KNOWLEDGE MANAGER *for illustration, we will explain in detail what the problem is, how it can, in principle, be solved by means of CBR, and why the model of Case Retrieval Nets appears to be particularly useful for this type of application.*

As in Chapter 8, we conclude this chapter by presenting more briefly related projects.

9.1 Basic Ideas of Knowledge Management

In today's world, knowledge is one of the most valuable assets of modern companies. This is in particular true for highly industrialized countries with high wages and the need to achieve industrial growth by innovation and additional services which could, in the required quality, not be provided by less industrialized countries. To make use of this asset, a company-wide knowledge management (KM) is required which has to cover the entire production process, including market analysis, product design, manufacturing, sales, and after-sales customer support.

In general, KM is concerned with the handling of knowledge existing in a company, organization, or group of people in order to facilitate creation, access, and reuse of that knowledge. Its primary goals are

- to gather information existing somewhere in the organization,
- to create valuable knowledge from this information, and
- to provide access to this knowledge.

Very briefly, knowledge management

> *"is a process of converting knowledge from the sources accessible to an organization and connecting people with that knowledge"* (O'Leary 1998a).

Today, knowledge management is a highly important issue in most companies for a number of reasons:

- Firstly, globalization forces companies to operate with offices in different countries and even continents. The traditional exchange of information and knowledge is hindered by this geographic dispersion.
- Secondly, modern IT, first of all the wide-spread availability of Internet technology, supports the utilization of distributed knowledge sources but requires some means for managing the information overload.
- Thirdly, there is a need for converting the vast amount of information that can be collected into valuable knowledge, i.e. into knowledge that the company or organization can make profit of.

To pay sufficient attention to KM tasks, about half of the major firms have appointed a *Chief Knowledge Officer* (CKO), according to O'Leary (1998a).

The most important requirement for a successful knowledge management strategy is that well-defined business processes are established which allow for an efficient exchange of information and guarantee that knowledge is available whenever needed. Apart from these business processes, however, modern information technology can provide useful tools to support various aspects of KM.

9.1.1 Aspects of Knowledge Management

With respect to the sub-tasks that a KM system may fulfill, one may distinguish the following activities of knowledge management:

Knowledge Harvesting is concerned with identifying valuable knowledge and collecting it such that others may profit from this knowledge.

Knowledge Discovery refers to processes of inferring valuable knowledge from so far knowledge-poor data, for example by data mining processes.

Knowledge Dissemination is concerned with processes of distributing knowledge and providing the (technical) infrastructure such that it may be accessed whenever needed within an organization.

Knowledge Reuse is related to making use of prior experiences when solving new tasks such that re-inventing the wheel is avoided.

Knowledge Preservation, finally, is concerned with preserving the knowledge that exists in some organization over a longer period of time, for example when experienced staff retires.

Depending on the source of knowledge, various activities are applicable (O'Leary 1998a; O'Leary 1998b):

- If the knowledge sources are the individuals working in some organization and the objective is to make their personal knowledge and experiences available to a wider group within the entire organization, then *knowledge harvesting* and *knowledge reuse* play a primary role.
- If data in the form of databases has to be analyzed, then techniques of *knowledge discovery*, such as data mining, should be applied.
- If knowledge is contained in textual documents, then also *knowledge discovery* techniques, for example by performing Information Extraction (Cowie and Lehnert 1996), as well as *knowledge reuse*, as traditionally by means of Information Retrieval (Salton and McGill 1983; Rijsbergen 1979), are of crucial importance.

Knowledge preservation, and *knowledge dissemination* are aspects that have to be addressed by any KM system, independently of the knowledge source.

In this chapter, we will address the problem of managing the knowledge contained in documents. For this, we will clarify what we consider typical *know how* documents and describe how Case-Based Reasoning techniques could be applied to perform a content-oriented search in such documents.

9.1.2 The Knowledge Contained in Documents

In modern industry, a widespread use of data collections can be observed. Virtually anything that can be recorded is being stored in databases. So-called *Data Warehouses* are widely used and provide information about all processes, products, orders and the like.

When it comes to preserve knowledge and experiences, however, human beings very much prefer to express themselves in natural language rather than being restricted to a rigid data format. This has several reasons:

- Firstly, formal languages, which are required for storing information in databases, normally do not provide sufficient flexibility and expressiveness.
- Secondly, if being sufficiently expressive, these languages are hard to handle in particular by a non-expert user who is unwilling to learn a formal language just to query some kind of information system.

Consequently, a huge amount of knowledge is stored in natural language, i.e. in the form of textual documents. Examples for such experience bases are

- collections of Frequently Asked Questions;
- news group files;
- handbook, manuals, and program documentations;
- informal notes.

In the following, we will refer to these documents as *know how* documents. When comparing such documents to texts in general, they show a number of specific properties:

1. These documents will in most cases discuss problems related to a specific domain.
2. Major parts of these documents are given as natural language text.
3. In addition to the text, however, the documents typically also contain structured data.
4. Last but not least, these documents usually have some kind of pre-defined internal structure.

Example 9.1 If a user visits the WWW site of some hardware vendor and reads the FAQs, then it is obvious that these FAQs will describe problems and solutions related to the products of the vendor. The texts contained in the question and answer sections of the FAQs will describe problems and their solutions, respectively. Apart from this text-based representation, the FAQs will also contain product names, version numbers, operating systems and so on which are best encoded in an attribute-value representation. Finally, the FAQs will have a question, an answer, and possibly a title.

To summarize, *know how* documents show some specific properties which allow them to be called semi-structured and which clearly distinguish them from other textual data, such as novels or WWW pages in general.

Furthermore, *know how* documents usually contain substantial parts of the knowledge assets of a company. Of course, this knowledge is much harder to manage than, for example, product specifications contained in a database where a well-defined logic can be applied to retrieve the relevant information.

Traditionally, techniques from Information Retrieval (IR, Salton and McGill 1983; Rijsbergen 1979) or even a simple keyword search are applied for this task, such as in the widely used Internet search engines. These approaches have a number of shortcomings which are due to the fact that they provide generic techniques which are applicable in virtually any domain. On the one hand, this allows for a wide-spread usage. On the other hand, domain-specific knowledge can hardly be integrated into such approaches. For example, knowledge of the following kind can hardly be considered in such a system:

Domain specific concepts and meanings associated to terms:
When speaking about printer problems, the concept `jam` clearly describes a problem with the paper feeder and thus disambiguity is avoided.

Relationships between concepts:
Two different printers may be highly similar because they are both ink-jet printers; on the other hand, they substantially differ from any laser printer.

Structure of the domain:
Printer problems can usually be divided into installation problems, printing problems and the like.

Structure of documents:
FAQs on the WWW site of a printer manufacturer will most likely have the above discussed title-question-answer structure.

9.2 Textual CBR for Knowledge Management

Although CBR has its roots in cognitive psychology, most CBR systems so far have been built in environments where cases can be represented by means of more structured data, such as feature vectors, graphs etc. But, given that people prefer to use natural language for storing their experiences or querying an information system, and that Case-Based Reasoning is concerned with reusing prior experiences for problem solving, the question arises how CBR can deal with *know how* documents as described above.

9.2.1 Methodological Differences to Other Areas

When comparing this type of application to other areas, a number of differences become obvious which will influence the design and implementation of a related system. We will explain these problems in the following; possible solutions will be suggested when the SIMATIC KNOWLEDGE MANAGER is described in subsequent sections.

Unstructured Cases

As mentioned above, cases will be much less structured. In the extreme situation, a case will no longer be representable by means of a set of features and associated values but by just a single feature (namely *text*) having a (large) set of values. This, obviously, has severe consequences for the techniques that can be used within such a system. For example, decision-tree oriented approaches for case retrieval are not appropriate.

Size of Documents

Related to the above point is that the size of the utilized documents may differ significantly. In the case of structured data, too, *missing values* may occur but usually a kind of *typical* case size can be determined. This is not true for *know how* documents where some documents may be larger than others by several orders of magnitude. This implies difficulties for constructing the similarity measure in such areas.

Ambiguity of Representations

Due to natural language being used in documents, problems arise with respect to the notion of information entities. In particular, it may be difficult, if not impossible, to satisfy the requirements on representations as discussed in Section 3.2.1. This is due to two problems known from research in Information Retrieval, namely the *ambiguity problem* and the *paraphrase problem*. The former refers to the fact that a single word or phrase in natural language may refer to completely different things (depending on context). The latter, in contrast, is concerned with the observation that completely different natural language phrases may describe exactly the same fact in the real world.

When properly designed, such situations should not occur in a system based on structured data. That is, a concept like `Color=Red` should be unique and unambiguously refer to a property of the objects in the domain. Even when using more complex, e.g. object-oriented, case representations, this requirement still holds.

9.2.2 Knowledge Containers for Textual CBR

One possible way of approaching the design of a system for Textual CBR is by considering the knowledge containers introduced by Richter (1995) and already described in Section 2.2.5. These are:

(a) the *vocabulary* used to describe cases;

(b) the *similarity measure* used to compare cases;

(c) the *case base* containing all the stored cases;

(d) the *adaptation knowledge* required for transferring solutions.

In the context of managing *know how* documents, we will assume that documents are available which can serve as a starting point in the construction of the case base **(a)**, and that adaptation knowledge **(d)** will be of no use here since the task is to *retrieve* semantically related documents.

In that sense, the goal is to construct a decision support system as discussed in Section 1.1.2 rather then to implement fully automatic problem solving. Hence, the Reuse step of the CBR cycle (see Section 2.2.4) does not necessarily have to be performed within the CBR system itself.

Consequently, there are three important questions to be addressed:

1. How to determine an appropriate vocabulary for case representation?

2. How to (automatically) converted documents into cases?

3. How to assess similarity of cases?

Vocabulary

For identifying the vocabulary used, a careful knowledge acquisition is required during which features important in the particular domain, specific terms, names of devices and products etc. have to be identified. The result of this process is a *concept dictionary* containing all the relevant terms including their contexts.

Note that for the construction of this dictionary, not only statistics about keywords may be considered but also techniques from Natural Language Processing and virtually any kind of information available for the particular domain (e.g. product databases). In Section 9.2.3 we will go into more detail about where this knowledge may come from and how it can be represented. For now let us assume that a properly constructed *concept dictionary* is available and that we can consider each entry in that dictionary as an *information entity*.

From Documents to Cases

Given this *concept dictionary*, a parser can be implemented which automatically extracts the concepts contained in a text and builds a case representation of the document.

Note that this *parsing* process is not limited to recognizing simple keywords. Instead, a number of more complicated tasks are performed, such as recognition of multi-word expressions, Information Extraction (Cowie and Lehnert 1996; Cunningham 1997; Riloff and Lehnert 1994), and recognition of higher level concepts (Glintschert 1998).

As each of these concepts is represented by an IE, the case originating from a document can be represented as a set of IEs as a result of this parsing process.

Similarity of Textual Cases

Once cases are represented as sets of IEs, similarity of cases can be determined based on these IEs. However, similarity assessment is not limited to considering, for example, how many IEs two cases have in common. Rather, knowledge about the relationships of the concepts is taken into account. If this knowledge is *compiled* to a similarity function *sim* describing the degree of similarity between any two elements of the vocabulary, then the similarity SIM of two documents d_1 and d_2 can be computed by

$$SIM(d_1, d_2) = \sum_{e_i \in d_1} \sum_{e_j \in d_2} sim(e_i, e_j)$$

and applying a standard scheme of normalization, for example based on the size of the documents.

9.2.3 Knowledge Representation via Knowledge Layers

In the descriptions above, we assumed that a properly constructed concept dictionary is available based on which documents can be converted to cases.

As we want this concept dictionary to contain the knowledge relevant for the particular domain, the question arises where this knowledge comes from and how it should be represented in a Textual CBR system. For this, we will in the following describe *knowledge layers* (Lenz 1998a) as containers for keeping both, knowledge about the vocabulary used in a domain as well as similarity relationships. After that, *knowledge sources* (Lenz 1998c) are described which may be used to fill the *knowledge layers*.

According to the characteristics of the various terms used in *know how* documents, it appears to be advantageous to separate the knowledge in several *knowledge layers* each of which is responsible for handling specific types of knowledge that may occur in the documents:

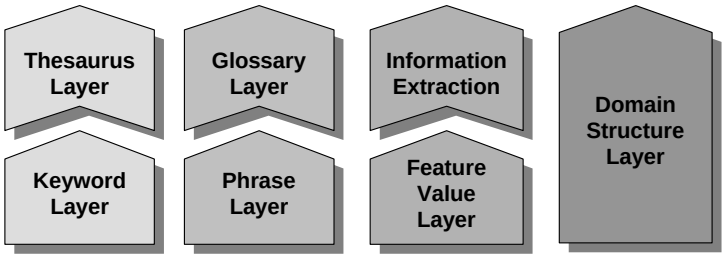


Figure 9.1: Knowledge Layers for Textual CBR

Keyword Layer: contains some kind of keyword dictionary which is used for recognizing simple keywords, for ignoring stop-words etc.;

Phrase Layer: contains a dictionary of domain-specific expressions used for recognizing more complex phrases not normally used in general-purpose documents, such as names of modules and devices;

Thesaurus Layer: contains information about how various keywords relate to each other in terms of (linguistic) similarity;

Glossary Layer: contains information about how elements of the keyword and phrase layers relate to each other in terms of domain-specific similarity;

Feature Value Layer: contains a set of features and their values as they might occur in the specific domain, such as names and release numbers of operating systems, physical measures etc.;

Information Extraction Layer: contains an *Information Extraction* module which is used to automatically extract structured information, feature values etc. from the textual descriptions;

Domain Structure Layer: contains a description of the domain structure allowing some *clustering* of documents, an example would be the distinction of printers in laser and ink-jet printers.

Note that in the above order the knowledge layers become more and more knowledge-intensive; also some layers require the existence of others as sketched in Figure 9.1.

As described above, each knowledge layer can be seen as a component storing knowledge in a *declarative* manner. To actually utilize this knowledge, each layer is enhanced by means of a specific functionality allowing to access that knowledge. For the keyword and phrase layers, for example, specific parsers are required which are able to recognize the concepts in natural language texts.

In terms of the knowledge container model, the keyword, phrase, and feature value layers belong to the terminology of case representation **(a)** whereas the remaining layers describe a piece of the similarity measure **(b)**.

9.2.4 Knowledge Sources

After having established the above layers as a desired architecture for encoding knowledge of a Textual CBR system, the question arises as to how each of these layers can be filled for a specific application. In general, a careful knowledge engineering process is necessary for this purpose and each layer requires specific procedures and techniques.

Keyword Layer

As mentioned above, the keyword layer basically consists of a keyword dictionary with an associated keyword parser using this dictionary for recognizing simple expressions in the documents. For building the keyword dictionary, techniques known from the Information Retrieval community (Salton and McGill 1983; Lewis and Sparck Jones 1996) can be used, such as:

- statistics about frequency of terms;
- stop-word lists;
- information on stemming of words.

To go beyond these statistical techniques and include linguistic knowledge, we also used *part-of-speech tagging* to obtain information about semantic categories of words as well as more advanced stemming information. This is particularly useful as some of our applications deal with German texts and we, thus, have to cope with an even more irregular grammar.

Phrase Layer

For the phrase layer, too, a parser is required which runs on a defined phrase dictionary and recognizes domain-specific phrases and a given document. The major differences to the above keyword layer are that

- phrases can not normally be obtained in sufficient quality by pure statistical analysis of a document collection;
- recognizing phrases is more complicated than parsing simple keywords because parts of a phrase may occur separated in a sentence;
- the keyword layer may be reused in other applications whereas the phrase layer is highly domain-specific.

To construct the phrase dictionary, application-specific knowledge sources have to be used, such as

- documentations and manuals of (software) products from which names of modules, components, menus etc. can be extracted;
- product catalogues containing detailed descriptions of products, releases, versions etc.
- more general dictionaries containing expressions not specific to a particular application but to some application area; for example, FOLDOC[1] provides a glossary of computer science terms.

During the knowledge acquisition process, these knowledge sources have to be scanned for relevant terms, i.e. for terms and phrases that might occur in the documents. In this phase, the integration of domain-experts is essential. Furthermore, a more advanced parser is required which is able to recognize phrases in the texts.

[1] The Free On- line Dictionary of Computing http://wagner.princeton.edu/foldoc/

Thesaurus Layer

The task of the thesaurus layer is to relate different keywords to each other. For example, information about synonyms, more abstract, and more specific terms appears to be highly useful.

For this, a general purpose thesaurus can be used, such as WordNet[2] (Miller 1995). If available, such a tool can be used for extracting various types of relations between a given pair of keywords and to assign a certain similarity value depending on this relation.

Unfortunately, WordNet is currently available for English only whereas some of our projects have to deal with German texts. To obtain a certain amount of thesaurus information for these applications, we utilized the fact that German texts very often use composite nouns. From these, an abstraction hierarchy can be derived semi-automatically. More precisely, composite nouns can be used to automatically construct lists of related terms which afterwards have to be scanned and checked manually. Based on our experiences, about 80% of the relationships derived this way are, indeed, meaningful.

Glossary Layer

Similarly to the thesaurus layer, the glossary layer is used to relate application-specific terms to each other by specifying similarity relations. Major differences to the thesaurus layer are that application-specific terms will hardly be contained in general purpose thesauri and even if they are, the actual meaning, or relations, between two terms may change for a specific application.

Consequently, an application-specific thesaurus has to be built for which glossaries and other sources in which terms are explained can be utilized. To a large extent, knowledge engineering techniques are required here, i.e. in cooperation with domain experts a model has to be built which relates application-specific terms to each other and thus provides the basis for a similarity measure.

Feature Value Layer

The feature value layer is designed to contain information about knowledge items that are best represented by means of traditional attribute-value pairs. These can be obtained by discussing with domain experts which attributes are normally used to describe the products and services of the application at hand. Also, an existing document collection can be scanned to see which feature values actually occur.

According to our experiences, such feature values are often separated from other parts of the documents in that special sections are used to encode these. Also, techniques from Information Extraction, in particular *Named Entity Recognition* (Cunningham 1997) can be applied to automatically scan a document collection for feature values.

Given the knowledge about relevant feature values one has to consult the domain experts again in order to construct an appropriate similarity measure which should include weightings of the various features as well as descriptions of similarity between the different feature values.

Information Extraction Layer

Though documents often contain a special section from which feature values can be obtained, sometimes the textual parts of documents also contain expressions which should better be encoded in terms of a more structured representation, such as attribute-value pairs. Also, queries posed by users normally lack an appropriate structuring.

Consequently, a certain amount of structural information should be extracted directly from the texts. As mentioned above already, Information Extraction techniques (Riloff and Lehnert

[2] http://www.cogsci.princeton.edu/~wn

1994) can be utilized for this task. In particular, one may scan the textual components for specific triggers and try to recognize additional information in the form of attribute-value pairs. These techniques not only allow for recognizing feature values in a textual description but can also be used to represent the texts by means of higher level concepts (Glintschert 1998).

Domain Structure Layer

The domain structure layer can be considered as a special feature value layer in the sense that very often particular features exist which classify documents with respect to some application area, or *topic*, they belong to. This can be a very useful information in particular if there are areas which appear to be disjoint. Using a classification based on such a *topic*, entire sets of documents can be safely ignored and thus precision can be improved greatly.

Knowledge Acquisition Effort

Filling all the above knowledge layers obviously requires considerable effort once a particular Textual CBR system has to be built. Fortunately, some parts of knowledge acquisition can be reused for a number of applications whereas other parts remain highly domain-specific. In particular, the keyword and the thesaurus layer should be reusable across domains as long as the language of documents remains the same. According to our experience, the effort for filling these two layers is most time-consuming due to the vast number of terms involved.

All other layers, however, are highly domain-specific and, hence, require additional effort for each new application. Consequently, the question arises whether there is any benefit from this additional effort with respect to the performance of a system. In Section 9.4 we will evaluate the approach of Textual CBR for the SIMATIC KNOWLEDGE MANAGER and show that there are clear advantages of Textual CBR due to the knowledge that can be incorporated into the system.

In certain situations it may be promising to attempt to learn relationships of terms directly from the user of a Textual CBR system. A possible approach might be, for example, to utilize *user profiles* or to consider some kind of feedback in order to adjust the internal structures. Whether these approaches are feasible in the described environments is an open question. A problem might be that we expect a highly heterogeneous user group in which it may even be impossible to identify individuals.

9.2.5 The Hotline Scenario

The use of *know how* documents appears to be particularly useful in so-called *help desks* and *hotlines* that are installed in many companies in order to help customers having problems with products and services purchased from that company.

Today, more and more companies are installing such hotlines and a reliable and efficient customer support is of increasing importance when it comes to market shares. The reason for this is that maintenance costs often exceed the initial value and thus become a decision criterion of customers (Lenz, Burkhard, Pirk, Auriol, and Manago 1996). This is true both, for industrial equipment as well as for highly complex software systems.

In general, hotline applications can be characterized as follows:

- A hotline will always focus on a specific domain, e.g. on some type of technical devices, on a set of software components etc. This implies that no topics outside this domain will be dealt with. Also, a number of highly specific terms will be used such as names of components, functions and modules.

- The term *hotline* does not mean that customer enquiries have to be handled within seconds. Rather, finding the answers to problems usually involves a complex problem analysis and

thus may take some time. Nevertheless, questions concerning problems that have been solved before should, of course, be answered rapidly.

- Naturally, a hotline will rely very much on textual documents such as FAQs, documentations, and error reports. Also, the customers' queries will be given in free text plus some additional information such as names of hardware components, software release numbers etc.

Consequently, there is an urgent need for techniques which can support the hotline staff in retrieving relevant documents in a specific problem context. In particular, this seems to be a promising area for Textual CBR as hotlines primarily deal with *know how* documents as defined in Section 9.1.2 (Lenz and Burkhard 1997b; Lenz and Burkhard 1997a). In the following, we will give a detailed example of such an application.

9.2.6 The CBR-Answers System

CBR-Answers is a system developed in cooperation with *TecInno* which is particularly suitable for the above described hotline scenarios. That is, by means of CBR-Answers *know how* documents can be searched in specific problem contexts for relevant information that might be helpful for solving the problem. CBR-Answers makes use of the above described knowledge layers in order to represent the knowledge of a specific domain.

However, CBR-Answers is not a stand-alone system in the sense that it is directly applicable to a new domain. Rather, it requires customization with respect to interfaces, data structures, and the knowledge layers. The system does, nevertheless, provide all the functionalities for implementing a running system once the mentioned application-specific information is provided.

We will omit a detailed presentation of the overall system here as most of the components will be described in detail in the context of the SIMATIC Knowledge Manager.

9.3 The SIMATIC Knowledge Manager

The SIMATIC Knowledge Manager (SKM) is the result of a project performed in cooperation with *TecInno* and *Siemens* AG. More precisely, it is a product developed for the customer support group of *Siemens Automation & Drives*.

9.3.1 Description of the Domain

Siemens is selling a wide range of automation systems within its SIMATIC program world-wide. Subsidiaries of *Siemens* as well as other companies are engaged in repairing and maintaining this equipment. To support technicians when trying to solve problems at the customer's side, *Siemens* has a hotline for customer support which answers telephone calls. Also, web pages are used increasingly for providing information to customers and technicians, such as updates of drivers or news about the latest products.

To further improve this service, *Siemens* considered using a Textual CBR approach for supporting hotline staff and evaluated a prototypical implementation for a period of 3 weeks. This evaluation revealed that there would have been only a limited benefit when installing such a system directly for the hotline staff. The reason for this is that *Siemens* has indeed highly qualified staff working at the hotline who would only consider using a tool in about 30% of all requests (Busch 1998a). Furthermore, these questions most often described real hard problems which required a substantial amount of expertise and complex problem solving. Hence, these problems are also hard to solve by an IT system in general and one could not expect Textual CBR to perform very well for these problems.

However, as a result of this evaluation the scenario of an *automatic hotline* has been developed. The idea here is that external technicians are referred to a set of FAQs and related documents before they consider calling the hotline. In order to efficiently find related information in the huge amount of documents available, Textual CBR can be used for implementing an intelligent search engine working on textual documents provided by the hotline.

A major objective of the automatic hotline on behalf of *Siemens* is to achieve a *call-avoidance*, i.e. to avoid that hotline staff is bothered again and again with problems that have been solved before. Instead, the hotline should only be contacted in case of truly difficult, previously unsolved problems.

But the customers of *Siemens*, that is the technicians using the automatic hotline, would also benefit from the system:

- Firstly, the telephone hotline is usually very busy. Hence, calling technicians will be queued and the answers to problems may be delayed considerably.
- Secondly, in contrast to the the telephone hotline, Internet services are available 7×24.
- Thirdly, the telephone hotline is not free of charge but rather a pre-paid *SIMATIC card* is being used to charge for the services provided. The Internet service of an automatic hotline may be offered to the technicians for a much lower fee or even free of charge.

Consequently, the SIMATIC KNOWLEDGE MANAGER has been implemented for *Siemens* which uses Textual CBR techniques and provides access to relevant information contained in documents published by *Siemens* according to the automatic hotline concept.

9.3.2 Motivation for a Textual CBR Approach

Techniques of Textual CBR as described in Section 9.2 have been used for building the SIMATIC KNOWLEDGE MANAGER because of several reasons:

1. Documents in the sense of *know how* documents (cf. Section 9.1.2) have been available. Also, more and more of such documents are being provided on the Internet.
2. Understanding the contents of these texts and finding related documents requires a substantial amount of knowledge, such as a domain-specific terminology, knowledge about the structure of the domain and about the various products. Hence, standard IR techniques are not applicable (cf. Lenz 1998d for a discussion).
3. Textual CBR directly works on the given documents, i.e. a conversion to cases is performed automatically inside the system. In contrast, other approaches, such as tools developed by *Inference*, require an explicit *case authoring*, that is a case base has to be acquired and maintained *in addition* to the original data. This, obviously, causes a tremendous overhead as well as consistency problems (Busch 1998a).

9.3.3 Advantages of CRNs

As described in Section 9.2, cases will be representations of documents consisting of sets of concepts. Hence, the ability of Case Retrieval Nets of handling cases which are sets of information entities is a major advantage in the domain of the SKM.

In particular, a composite similarity measure according to Definition 5.5 will certainly not suffice as the main parts of the cases will be representations of the textual components of the documents rather than an attribute-value-based representation. Consequently, a number of more traditional methods for case retrieval, such as kd–trees (Wess, Althoff, and Derwand 1993) are not applicable.

Also, a hierarchical structure as in MOP hierarchies (Riesbeck and Schank 1989) cannot be used. Even worse, it seems difficult, if not impossible, to apply any kind of *top-down* oriented retrieval as there are no criteria which could be used for *rejection* of cases as discussed in Section 5.3.4. Rather, the idea of *accumulating* information for similarity assessment, as implemented by CRNs, is much more appropriate.

Efficiency of Case Retrieval Nets has, of course, also been a decision criterion as a huge number of documents can be expected once such a system is installed.

9.3.4 Domain Analysis

According to the scenario of an automatic hotline, the description of the domain according to Section 7.3.1 is straightforward:

Domain Objects

The *domain objects* are, of course the text documents existing at *Siemens*. More precisely, a set of document collections exists each of which has a certain focus with respect to the objective of the documents. Currently, there are

Frequently Asked Questions containing question-answer pairs for problems solved by the hotline;

User Infos providing up-to-date information about new versions of products;

Download Infos describing availability of driver updates etc. which may be downloaded directly from the WWW;

News containing descriptions of new products and their properties for company-internal purposes;

Problem Books containing detailed descriptions of problems that occurred and how these can be overcome;

Problem Reports listing recently observed problems and errors without detailed information and usually without hints for solving these.

The first three of these document types are public, i.e. may be accessed by external users. Only company-internal users are permitted to access the last three document types. Furthermore, each of these document types has its own specific structure in the sense of sections that usually are part of the documents, such as QA-pairs in FAQs.

Domain Model

The *domain model* is much harder to specify. Obviously, it should capture information about the various document types and their structure, about the terms being used in the documents, and about the overall structure of the domain. In fact, the *knowledge layers* as described in Section 9.2.3 should be used for modeling the domain.

A number of knowledge sources available in the company can be used to fill the knowledge layers. These include

- a structure of the domain as described by a domain expert,
- listings and descriptions of products from electronic catalogues,
- the expert's knowledge about relationships between various products,
- sample document collections which contain the relevant terms of the domain.

In contrast to the VIRTUAL TRAVEL AGENCY, this domain is not finite a priori. Rather, in each new document and in each query new terms may occur that are not contained in the *domain model*. Even more, by means of the information extraction layer, additional IEs may be constructed which represent specific feature values recognized in case documents. For a given set of case documents, however, all the knowledge layers described in Section 9.2.3 will be finite and, thus, the internal representation of the domain is finite, too.

Business Model

The *business model* for the SKM directly follows from the automatic hotline concept. There are three different groups of actors involved:

Customer Support Group: Members of the customer support group build up an initial document collection based on which the Textual CBR approach may run. Also, they are responsible for maintaining the system both in terms of case as well as similarity maintenance.

Hotline Staff: The hotline staff is involved in so far as they answer questions that could *not* sufficiently be answered by the SKM system. When answering, they will produce new documents, such as FAQs, which may be added to the system by the customer support group.

Users: The users, i.e. the technicians working in the field, may query the system when facing a specific problem with SIMATIC products. The document collections that are accessible to the user depend on whether it is a company-internal user or not. Users are not allowed to change the system in any way, also there is currently no means of *personalizing* it. Users will, however, provide feedback about the utility of the search results.

In contrast to the telephone services provided, users should for the present not be charged for using the SIMATIC KNOWLEDGE MANAGER although this policy might change later on.

Note that many of the business processes required for running the SIMATIC KNOWLEDGE MANAGER have already been installed at the SIMATIC hotline independently of the planned Textual CBR system.

Application Scenario

The *application scenario*, finally, has been defined according to the experiences gained during the evaluation period (cf. Section 9.3.1). The target user is not the hotline staff but rather the technician working with SIMATIC products, most often outside the company. Consequently, we can assume that the user has, on the one hand, a thorough understanding of the domain, the principle technology, and the products. On the other hand, s/he will not have complete knowledge about *all* the products and all failures that might occur. Also, the outside technician will most often lack the most up-to-date information.

The interface by which the user is enabled to query the system should take into account the focus on textual representations in the domain and, hence, permit natural language queries.

Table 9.1 summarizes the results of *domain analysis* by briefly answering the questions listed in Section 7.3.1.

9.3.5 System Specification

Cases, Case Bases, and Queries

As described in Section 9.2, a *case* is a representation of a single document by means of a set of concepts, or information entities.

Table 9.1: Answers to the questions of *domain analysis* for the SIMATIC KNOWLEDGE MANAGER application

1.1	existing document collections for SIMATIC products
1.2	available as HTML pages, semi-structured
1.3	updates if required, mainly insertion of new documents
2.1	taxonomy of domain, glossaries and product catalogues
2.2	general-purpose dictionaries and thesauri for knowledge layers
2.3	customer support group verifies and adjusts similarity model
3.1	SKM installed on SIMATIC WWW server, UNIX environment
3.2	documents provided by SIMATIC customer support
3.3	maintenance by customer support staff
3.4	technicians with access via WWW, free of charge
3.5	feedback from users about utility of retrieval results
4.1	technicians as well as company-internal users
4.2	users have a thorough understanding of the domain
4.3	limitations to external users w.r.t. provided document collections

A *case base* is the set of all cases for a single type of document. Hence, the SKM deals with 6 case bases according to the document types listed in Section 9.3.4.

In order to avoid inconsistencies and to ease maintenance, a case base should be implemented as a *view* on the original data as discussed in Section 7.3.2. That is, case bases may be used internally to the system if required but should not be present outside as an autonomous data structure.

A *query* is an information request by a user which consists primarily of a natural language description of the problem but may also contain structured information, such as version numbers or the sub-domain the problem belongs into.

Information Entities, Similarity Relationships, and Relevances

The *information entities* as well as the *similarities* and *relevances* of IEs have been obtained by filling the above described knowledge layers. To be more precise, the content of these layers has been obtained by the following processes (Lenz 1998a):

Keyword Layer: derived by analyzing a document collection, using statistics about term frequencies etc., plus linguistic tools (*part-of-speech* tagger) and tables to include various word forms;

Phrase Layer: filled by manually analyzing product catalogues available electronically;

Thesaurus Layer: mainly derived from analyzing German composite nouns plus manual insertions;

Glossary Layer: partly derived from product catalogues which also include a clustering of products, partly built by SIMATIC customer support staff;

Feature Value Layer: obtained from product catalogues and additional databases containing product descriptions, unique product and version numbers etc., also by analyzing document collections and searching for feature value patterns;

Information Extraction Layer: Information Extraction module directly built on top of feature value layer;

Domain Structure Layer: built in discussion with SIMATIC customer support staff.

The originally infinite domain is converted to a finite one by assuming that all terms can be ignored which are not represented as an IE in one of the knowledge layers.

9.3.6 System Implementation

Currently, the SIMATIC KNOWLEDGE MANAGER consists of a number of modules as shown in Figure 9.2. As described in the following, the system is accessible via the World Wide Web by standard HTML forms.

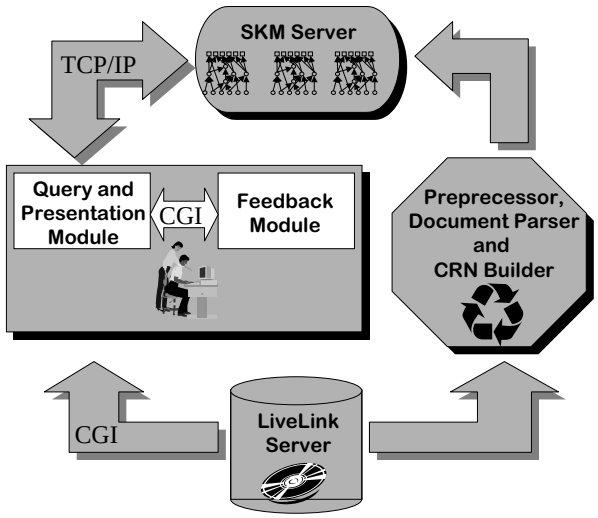


Figure 9.2: Architecture of the SIMATIC KNOWLEDGE MANAGER

Retrieval Server

Of course, the most important component is the *Retrieval Server* which communicates with the *Query and Presentation Module* via TCP/IP, performs the retrieval process for a given query and returns the result as a list of case identifiers.

As has been the case for the VIRTUAL TRAVEL AGENCY already, this server manages not just one but six Case Retrieval Nets — one for each type of document. As discussed in Section 8.2.6 this has the advantage of allowing updates separately for each CRN. Also, the different structure of the document types can be captured in a better way. Last but not least, this provides an easy means for implementing the limited access for external users as retrieval is simply performed on the three public data sets for these.

Query and Presentation Module

The *Query and Presentation Module* basically consists of an HTML form and a CGI script. For the user, it is an interface for entering an information request. For this, a textual description of the problem, the sub-domain the problem most likely belongs to as well as some technical information (such as product numbers) can be entered.

This data is sent to the *Retrieval Server* which returns a list of case identifiers. For the cases in this list, additional information is requested from the *Data Server*, such as a proper title and the URL of the original document. Then, this information is displayed in a form that

most users will be used to from standard Internet search engines and that allows to display the original document by simply following the links provided.

If the search did not yield results of sufficient quality and the problem of the user remains unanswered, then a so-called *Trouble Ticket* is released. This is an additional HTML form in which the user should provide more detailed information which is then forwarded directly to the hotline staff.

If the search was successful, however, a separate HTML form will be used to ask for feedback by the user. This feedback basically comprises a rating of the services provided and is evaluated regularly by the customer support group (Busch 1998b).

Update Module

The *Update Module* is used for reflecting changes in the document collections or the similarity model and to keep the *Retrieval Server* up-to-date. Updates will not be performed automatically but always started by the customer support group.

When updates are required, the given document collections will first be *pre-processed* such that all the various document formats are converted into a single format of *case documents*. These will then be parsed and converted to a set of case bases (one for each document type) which are, however, only used internally by the system. In fact, the case bases only exists as long as the update process is performed. After that, the newly constructed Case Retrieval Nets can be considered as an indexing mechanism directly to the original documents. In that sense, a case base in the SIMATIC KNOWLEDGE MANAGER can be seen as a *view* on the original data as discussed in Section 7.3.2.

Once the new Case Retrieval Nets have been constructed, the *Retrieval Server* is sent a message telling it to discard the old CRNs and to load the new ones.

As parsing the original documents requires a considerable amount of time, an *incremental* update may be performed later on. During this, only the modified (i.e. new, deleted, or altered) documents are considered while all others remain untouched. An additional internal structure assures that a complete Case Retrieval Net can be built based on the information from the modified documents and from the original CRN. This is possible due to the flexibility of Case Retrieval Nets with respect to maintenance as discussed in Section 5.3.3.

Data Server

The *Data Server* of the SKM currently provides two basic functions: Firstly, it maintains an internal structure which allows to directly access information required by the *Query and Presentation Module*, such as the title of a case or the URL of the original document. Secondly, it uses the LIVELINK system installed at *Siemens* to access and display the original HTML documents.

Further Implementation Details

The system described so far is, in fact, not a system developed and applicable for *Siemens* only. Rather, the SIMATIC KNOWLEDGE MANAGER is a customized version of the CBR-ANSWERS described in Section 9.2.6. The SKM is customized in so far as

- the elements of the GUI are specific for *Siemens*;
- a *session management* has been implemented in order to be able to trace the path of single users in the system;
- the knowledge contained in the knowledge layers captures specific know how about the SIMATIC domain;

- the original data is accessed via the installed LIVELINK system.

As CBR-ANSWERS in general, the SKM has been developed in cooperation with *TecInno*, Kaiserslautern.

In the above described server implementation, all components of the GUI are implemented as CGI scripts known from standard WWW programming (cf. Figure 9.2). Similarly to the VIRTUAL TRAVEL AGENCY, HTML *template* pages are being used to describe the graphical layout of all interfaces. Thus, the layout may be changed without any programming effort and the programs are not overloaded with too much information about the HTML layout. All components have been written in C++, the entire system consists of approximately 35,000 lines of code.

In addition to the WWW server, a CD-ROM solution has been built which addresses users that do not have access to the Internet. In principle, the same technology is applied here but the search is performed *locally*, that is based on the documents that are stored on the CD-ROM. An integration of the local and the Internet search has been implemented which combines fast access over the CD-ROM with up-to-date information from the WWW.

9.3.7 System Maintenance

Maintenance of the SIMATIC KNOWLEDGE MANAGER has to be considered with respect to two components:

- Firstly, the document collections underlying the system may change.
- Secondly, changes to the knowledge layers may require an update in order to reflect modifications on the similarity model.

If required, an update process will be started by the customer support group. The update itself is performed automatically based on the specified document collections and the similarity model expressed by means of the knowledge layers.

9.3.8 Current State of the SKM

The first version of the SIMATIC KNOWLEDGE MANAGER, available to internal users only, had been installed at *Siemens* in March 1998. The Internet version[3] went public in June 1998. At that time, the system handled approximately 7,500 German documents A first CD-ROM had been shipped in April 1998 at the Industrial Fair in Hanover. Figure 9.3 shows a snapshot of that system.

Recently, a number of improvements and extensions have been integrated into the system, including support of English documents and an even closer integration into the infrastructure at *Siemens*. In December 1998, version 3 has been released.

9.4 Evaluation of Textual CBR

The quality of the results obtained by a Textual CBR system is, of course, much harder to evaluate than, for example, for the VIRTUAL TRAVEL AGENCY. As shown in Section 5.3.1, a BCRN directly implements the computation of a similarity measure. Hence, the questions of completeness and correctness of the retrieval procedure do not arise in such applications; rather failures in the modeled similarity model have to be blamed for wrong retrieval results.

The situation is slightly different in Textual CBR. In these areas, one can never expect to have a complete and correct model of all the relationships due to the problems associated

[3] http://www.ad.siemens.de:8080/skm/html_00/ for the German version
http://www.ad.siemens.de:8080/skm/html_76/ for the English version

Figure 9.3: Snapshot of the SIMATIC KNOWLEDGE MANAGER shipped on CD-ROMs to clients of *Siemens*. As displayed, the system takes into account knowledge about the domain such as when mapping from `y2k` to the year–2000–problem.

with the use of natural language. As discussed in Section 9.2.3, a considerable amount of knowledge engineering is required even for an approximate model. This effort should be justified by improved retrieval performance compared to, for example, standard Information Retrieval techniques. In particular, the question arises how much each of the defined knowledge layers contributes to the overall retrieval performance.

In the following, we will empirically evaluate the results obtained by means of Textual CBR as implemented for the SIMATIC KNOWLEDGE MANAGER. We will show that the knowledge provided to the Textual CBR approach, indeed, improves the system's retrieval performance.

9.4.1 Evaluation Methodology

Measures for Evaluation

For evaluating the performance of a Textual CBR system, measures similar to *precision* and *recall*, known from the IR community, are a first choice. However, there are some crucial differences with respect to the underlying assumptions:

Firstly, measures like precision and recall, as defined, for example, by Salton and McGill (1983) are based on sets. In practical CBR systems, a set of cases will be retrieved, too, for example when searching for the k best cases. This set, however, will be ranked and this ordering is not taken into account by the usual notions of precision and recall.

Secondly, both measures assume that for a set of queries *relevance judgments* are known. For evaluating the SIMATIC KNOWLEDGE MANAGER, this appears to be a major difficulty as the SKM deals with a highly specialized domain where relevance judgment is possible only by a domain expert. What's more, one can easily imagine that it is hard, if not impossible, to get these experts perform a task which is mainly interesting from an academic point of view.

Thirdly, *recall* originally measures the percentage of the retrieved relevant documents with respect to *all* relevant ones. In the hotline scenario, however, the goal is not to retrieve *all*

relevant items but rather to answer a query successfully. Hence, *one* relevant document is sufficient if it helps solving the problem (for a similar discussion see Burke, Hammond, Kulyukin, Lytinen, Tomuro, and Schoenberg 1997b).

Consequently, we have to modify these measures in order to make them applicable for our purposes. As the result of the retrieval process will be an *ordered* list rather than a set, precision and recall have been defined in such a way that they do reflect the ranking. Concerning recall, we can utilize the fact that (in accordance with the third remark) most documents appear to be unique in the sense that each document answers a specific query and there is hardly ever another document that answers this query. Consequently, we can keep the original notion of recall. We have to modify the notion of precision, however:

Definition 9.1 (Precision for Textual CBR) *If k documents have been retrieved for a query for which $r \geq 0$ relevant documents exist, then* precision *P is defined as*

$$P = \begin{cases} 1 & \text{if } k = 0 \\ \frac{r'}{k} & \text{if } k \text{ documents have been retrieved among which are} \\ & r' \leq r \text{ relevant documents} \\ \frac{r}{i} & \text{if all relevant documents have been retrieved and} \\ & i \leq k \text{ is the position of the last relevant document} \end{cases}$$

□

According to Definition 9.1, the value of precision will not further decrease once all relevant documents have been found but more documents are retrieved. This also corresponds to the way (human) users would utilize such a system as they would sequentially scroll through the list of documents but stop once an answer to the question has been found.

In addition, we will use a measure called *expected search length* (ESL) below. As in traditional IR, this denotes the number of documents that have to be searched until all relevant documents have been found. This measure, obviously, reflects in a much better way that the result of the retrieval process is an *ordered* list.

The problem concerning missing relevance judgements is harder to solve. To construct a set of queries for evaluation, we utilized the following observation: While relevance judgments can only be given by a domain expert, it is much easier to decide whether two queries have a similar semantics. Consequently, we randomly selected a set of FAQs, analyzed the contents of these, and paraphrased the question in several steps, from minor grammatical variations to complete reformulations. In the latter case, we changed as many expressions as possible but kept the names of devices, modules, components etc.

Example 9.2 An example of such a reformulation might be (remember that in the SIMATIC domain we have to deal with German texts, hence this example is only an approximate translation):

Original query:
In what modus do I have to shoot the 128 KByte EPROM for the CPU-944?

Modified query:
How do I have to run the 944 CPU for installing a 128 KB EPROM?

As this example shows, the semantics of the query may change slightly. But in any case, the answer to the original query will still answer the modified one. When applying this procedure, one knows which FAQs are relevant for each of the modified queries; as explained above it will be just one in most situations.

For the experiments described in subsequent sections, we randomly selected 25 FAQs from a set of 2,303 German documents contained in the release of December 1998 and paraphrased the questions. The original questions and their paraphrased versions are listed in Appendix A.

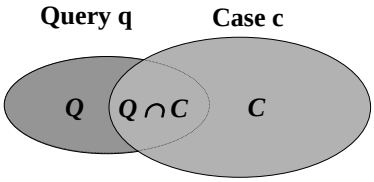


Figure 9.4: Normalization problem ilustrated

In most situations, just the original document appeared to be relevant but in 5 cases a second relevant document could be identified which described the same problem in a different context.

Evaluation of Alternative Normalization Methods

Compared to the domains described in Chapter 8, a crucial difference in Textual CBR is that the cases (i.e. the internal representations of documents) may differ significantly in size. Some FAQs, for example, only consist of a title, a question, and a one–sentence answer. Others may contain a longer explanation and, thus, be 2 or 3 pages in length (see also the discussion in Section 9.2.1).

As a consequence, the problem of *normalization* arises, i.e. if similarity of cases is based on common (or similar) IEs, how will IEs present in either the query or a case influence the degree of similarity. Figure 9.4 illustrates the problem: Here Q and C denote the set of IEs present in the query and the case, respectively, and $Q \cap C$ contains all IEs present in both (in this figure, similarity is not considered).

Of course, the ideal situation would be if $Q = C$, but in the very likely situation that such a case does not exists, the problem arises whether one would prefer for a given query q a case c_1 with additional IEs or a case c_2 with less IEs. In terms of the similarity measure, this concerns the way normalization is implemented: If similarity is computed based on achieved activations act of case nodes as always in CRN models, will similarity

(N1) be normalized according to the number of IEs in the query

$$\frac{act}{|q|}$$

(N2) be normalized according to the number of IEs in the case

$$\frac{act}{|c|}$$

(N3) be normalized according to both

$$\frac{act}{|q|} \times \frac{act}{|c|}$$

(N4) be set directly according to the activation act of c.

For the SIMATIC Knowledge Manager, method **(N1)** was chosen already during an early phase of system design because it seemed most appropriate for the given application scenario: A case fits best if the query is a subset of it and contains some additional information. Also, queries can be expected fairly small compared to cases (say one sentence in relation to half a page). In this sense, it illustrates again the notion of information completion as discussed in Chapter 3.

While this is only an informal argument, the evaluation revealed that method **(N1)** is, in fact, most appropriate. Figure 9.5 shows the precision–recall curve for the normalization

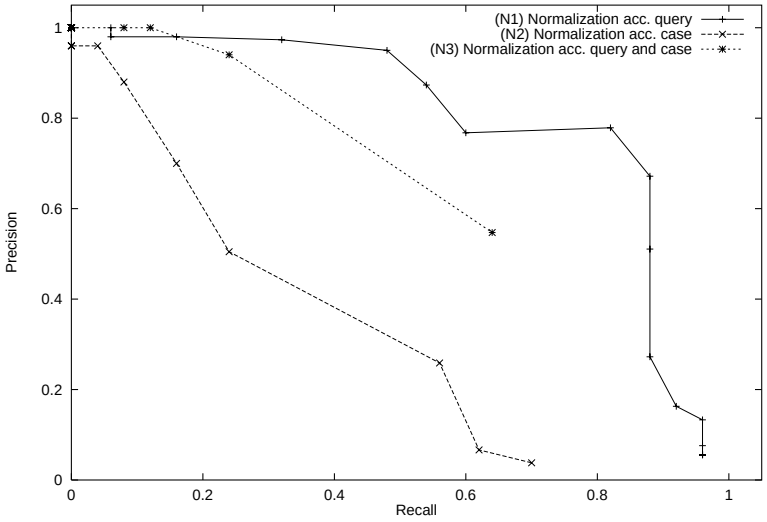


Figure 9.5: Precision–recall curves for the different normalization methods

methods **(N1)**, **(N2)**, and **(N3)**. Method **(N4)** is not shown since it will deliver exactly the same results as **(N1)**; this is because the query is the same for all cases and, hence, the ordering of cases depending on their activations would remain the same — only the actual similarity values would be scaled.

Evaluation of Knowledge Layers: An Ablation Study

To evaluate the contribution of each knowledge layer, we performed an ablation study by subsequently eliminating higher level knowledge layers. More precisely, the following experiments have been performed:

(E1) Keywords: Documents were retrieved based on a simple string matching of keywords. In fact, every word occuring in the documents except for a predefined list of stopwords has been considered as a keyword.

(E2) Keywords plus *topic*: In the SKM, a so-called *topic* is used to classify documents according to the domain structure. In this experiment, the *topic* (i.e. a part of the domain structure layer) was taken into account in addition to the keyword search.

(E3) Retrieval based on IEs: Documents were retrieved based on the IEs specified in the keyword and phrase layers.

(E4) Retrieval based on IEs plus *topic*: In addition to the IEs, the *topic* of documents was taken into account.

(E5) Full model: Finally, retrieval was performed using the full model, that is IEs (from the keyword and phrase layers), *topic* (from the domain structure layer), and similarities (from the thesaurus and glossary layers).

We did not

- distinguish here between the keyword and the phrase layer as both have been represented by means of a single dictionary and it did not make sense to artificially separate that dictionary;

- investigate the layers for feature values and information extraction because in the considered domain only very few pieces of information can be represented as feature values, and compared to an earlier study (Lenz and Burkhard 1997b) most of these have now been represented by means of standard IEs by the hotline staff.

Furthermore, one could think of yet other combinations of the knowledge layers and, thus, come up with further experiments. We think, however, that the above ordering somehow reflects the amount of knowledge engineering required for filling these layers. Obviously, a simple keyword search requires the least effort. Also, a kind of classification can be introduced in a fairly straightforward manner. In fact, some search engines on the WWW already do this. In contrast, the specification of IEs and even of similarities requires much more effort.

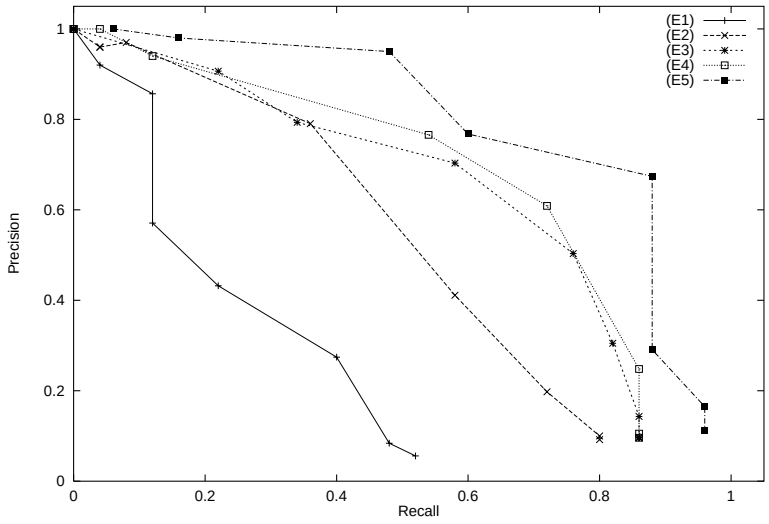


Figure 9.6: Results of the ablation study for the SIMATIC KNOWLEDGE MANAGER (see the text for an explanation of the numbers)

Figure 9.6 shows the results obtained from experiments by means of the usual precision–recall curves. The exact data is given in Appendix A. These results were obtained by subsequently lowering a similarity threshold during retrieval and thus allowing more documents to be contained in the result set.

Clearly, these results show that the knowledge acquisition effort that is required for building a Textual CBR system does, in fact, pay off:

- Compared to a simple keyword search, the use of a classification according to the *topic* of a document already leads to improved results.
- The utilization of IEs provides for another significant increase of performance. As this shows, the definition of IEs allows to capture the concepts contained in documents in a much better way than can be done with simple keywords.
- The utilization of similarity relationships further improves the results and allows for a nearly perfect recall with an acceptable value of precision.

Similar results are obtained if the ESL measure is used as displayed in Figure 9.7. Again, additional knowledge leads to an improved performance in that less documents have to be searched in order to find the relevant one(s).

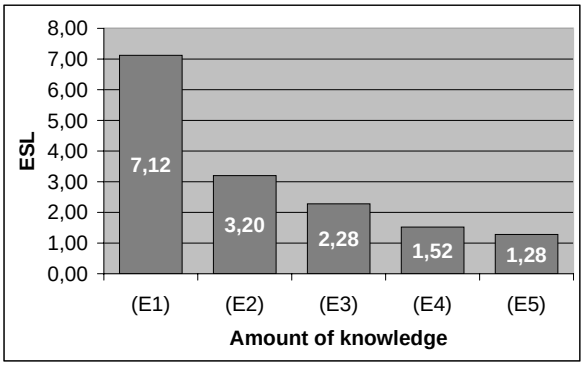


Figure 9.7: Expected search length depending on the amount of knowledge used in the SKM

9.5 Related Projects

Similarly to the SIMATIC KNOWLEDGE MANAGER presented above, we will now discuss two related projects in the area of Textual CBR. However, as both are similar to the SIMATIC KNOWLEDGE MANAGER to a great extend, we will only present the specific details of these applications here. In particular, we will not go into implementation details as both systems can be seen as customized versions of the CBR-ANSWERS system again. The major purpose of the descriptions is to illustrate the potential usage of the technologies introduced so far in slightly changed environments and based upon different types of documents.

9.5.1 FALLQ

Description of the Domain

LHS AG is market-leader in developing a customer care and billing system for cellular telephone service providers. In principle, *LHS* delivers just one, though highly complex, software product named BSCS. In practice, however, every customer (i.e. telecommunications company) demands some specific features which are not available for competitors.

Consequently, there are many different versions and releases which have to be maintained in parallel. Due to the rapid growth of the telecommunications industry, *LHS* is permanently employing new staff for the system development and customer support groups. Furthermore, a lot of distributed project teams work for different customer releases. Thus, some means of knowledge management is required which helps coping with the need of the market but avoiding repetitions of work and loss of knowledge and experience.

Example 9.3 It is often the case that highly similar requests by different customers are handled by different staff without knowing from each other. This happens, for instance, when fixing bugs.

On the other hand, *LHS* has a well-defined regime of documenting all processes (bug fix reports, customer requests etc.) in pre-defined electronic forms. Hence, it seemed straightforward to apply Textual CBR for making the knowledge contained in these documents available to other staff.

After the evaluation of a prototypical Textual CBR implementation (Lenz and Burkhard 1997b), *LHS* decided to apply this approach for *in-house knowledge management*. After the prototype had successfully passed internal testing processes, a first Intranet version has been installed at LHS in Summer 1997. The system currently handles approximately 45,000 English

documents of various types; negotiations about further improvements and extensions of the systems are going on.

Domain Analysis

The *domain objects* are the documents existing at *LHS* and describing major business processes. More precisely, a set of such document collections exists each of which is used to describe a specific process. All documents are written in English. Currently, handling of the following documents is implemented:

Frequently Asked Questions containing questions and answers for problems solved by the hotline;

Defects containing both known bugs as well as previous bugs and their fixes;

High Level Descriptions describing the functionality of certain modules on an abstract level.

Each of these document types has a specific content and structure which is defined by the management. For this reason, these can be considered as a kind of *form* that has to be filled out.

For building the *domain model*, some knowledge sources existing at *LHS* have to be utilized. These include

- knowledge about the structure of documents, relevant sections etc.
- names of modules, windows etc. as described in specific databases maintained by *LHS* to assure consistency in their projects;
- sample document collections which contain the relevant terms of the domain.

The *business model* at *LHS* is slightly different from the hotline scenario in so far as only internal users are allowed to enter the system up to now. Hence, there are just two groups of actors involved:

Customer Support Group: Staff of the customer support group run and maintain the system.

Users: Company-internal users may query the system when facing problems. In a later stage, one might even implement a *pro-active* approach in the sense that, for example when writing a document, the user is actively directed to related topics as expressed in similar documents.

Users are not allowed to change the system in any way, also there is currently no means of *personalizing* it.

The *application scenario* is specific for the above described domain: the target user group are the employees of *LHS* but not necessarily the staff of the customer support group only. For example, a member of a project development team may as well query the system to figure out what kind of bug may cause an observed unexpected behavior. In any case, every user can be assumed to have a thorough understanding of the domain in general and BSCS in particular.

Users should access the system via WWW interfaces over the existing Intranet. The interface by which the user is enable to query the system should take into account the focus on textual representations in the domain and, hence, permit natural language queries.

System Specification

Cases, case bases and queries can be defined as above for the SIMATIC KNOWLEDGE MANAGER. The major difference is in the contents of the knowledge layers which will then influence the IEs, similarities, and relevances. These knowledge layers have been filled as follows:

Keyword Layer: derived by analyzing a document collection, using statistics about term frequencies and such, plus linguistic tools and tables to include various word forms (e.g. Porter's algorithm, Porter 1980);

Phrase Layer: filled by analyzing documents with respect to multi-word expressions occurring often, plus databases available at *LHS* containing names of devices, modules, functions etc.;

Thesaurus Layer: based on a machine-readable thesauri, manually converted and corrected;

Glossary Layer: based on a glossary provided by *LHS*, manually converted;

Feature Value Layer: based on features that occurred in the document collection, *LHS* experts decided about inclusion;

Domain Structure Layer: currently not used

Information Extraction Layer: currently not used

Current State

The FALLQ system (Kunze and Hübner 1998; Lenz and Burkhard 1997b) is installed and in use at *LHS*. The system has been prototypical in so far as integration and maintenance issues have not been addressed sufficiently yet whereas the Textual CBR part has been fully implemented. By January 1999, a new version will be installed which provides all the functionality currently available within the CBR-ANSWERS system. This version will be an Intranet version first, Internet versions for clients of LHS will follow in 1999.

9.5.2 The ExperienceBook

Description of the Domain

The EXPERIENCEBOOK (Kunze and Hübner 1998) is a system developed for supporting system administration at Humboldt University Berlin. The main objective is to provide methods for building an *organizational memory* which integrates the knowledge of all members of the system administration group.

Domain Analysis

Domain objects, i.e. documents that could serve as a starting point for constructing a case base, had to be collected first when building this application. This differs from the above two applications. Documents were obtained by searching appropriate news groups as well as reusing the personal documentations of system administrators, such as notes and emails.

Building the *domain model* could, in contrast to the above two applications, not be based on well-defined knowledge sources available *within* the potential user group. Thus, external knowledge sources have been used to obtain at least a rough model. For example, the FOLDOC system mentioned above already contains valuable information relating special terms of computer science.

For the prototypical implementation, both the *business model* and the *application scenario* are quite simple: Members of the system administration group maintain the system by adding

new documents, and they query the system when facing problems. However, this acquired knowledge is shared as each system administrator also has access to the knowledge of the colleagues and may, thus, benefit from their special expertises in specific areas.

The similarity model is, at least for the prototype, maintained by members of the AI group which are actively involved in developing the CBR-ANSWERS system.

System Specification

Again, cases, case bases and queries can be defined according to the definitions given for the SIMATIC KNOWLEDGE MANAGER. The IEs, similarities, and relevances are derived from the specific knowledge layers of this application. The knowledge acquired during the FALLQ project could partially be reused:

Keyword Layer: built based on FALLQ keyword layer, extended and adjusted by analyzing document collections from news groups and such;

Phrase Layer: inclusion of FOLDOC terms, manually filtered and corrected;

Thesaurus Layer: utilization of WORDNET (Miller 1995), manually filtered and corrected;

Glossary Layer: currently not used

Feature Value Layer: added attributes such as operation systems and machine types;

Domain Structure Layer: currently not used;

Information Extraction Layer: currently not used;

Current State

The EXPERIENCEBOOK is fully implemented as another instantiation of the CBR-ANSWERS system. What is currently missing here, is an appropriate model and first of all a sufficient number of documents such that the system might provide at least a minimal benefit for the potential users. Consequently, the collection, integration, and maintenance of such data will be addressed in the near future.

Part IV

Discussion

Chapter 10

Related Work

One of the reasons why there is so much interest in CBR, may be attributable to the nature of CBR and how it closely resembles human reasoning.
Chris Lafferty, Product Manager, Internet Commerce Marketing, 3Com

---

In this chapter, we will discuss some other work and the relationships to the Case Retrieval Net model and to the applications presented earlier. For this, we will not only investigate other methods for case retrieval, but also present related work from Textual CBR, Information Extraction, and Knowledge Management. Finally, we will address some theories of cognitive science that appear to have a strong relationship to CRNs.

10.1 Comparison to Other Retrieval Methods

Since the retrieval process plays an important role in the CBR paradigm, a number of different techniques have been developed aiming at an improvement of both, accuracy and speed of the retrieval process.

In the following, we will briefly compare the ideas of CRNs to some of the major approaches to case retrieval. A more detailed comparison of these techniques, including analysis of complexity, performance benchmarks, and the ability to support specific similarity functions, can be found in (Goos 1994).

Independently of the utilized memory structures, all the approaches sketched below (except for FISH & SHRINK to some extent) share the idea that the case base has to be searched in a *top-down* manner as discussed in Section 5.3.4. In that aspect, Case Retrieval Nets differs from all of these other approaches. We will now go into some more detail.

10.1.1 Linear Search

Linear search is only applicable for small case bases as the effort grows linear in the size of the case base. In the worst case, the effort for retrieval in Case Retrieval Nets will also grow linearly as shown in Section 5.3.2. However, due to the distributed representation being used the effort will still be considerably lower than for a simple linear search. In most real-world situations, the similarity model will further *partition* the set of IEs in that only certain subsets have, in fact, a non-zero similarity. This further reduces the retrieval effort greatly.

Major advantages of a linear search, however, are that this method is easy to implement and highly flexible with respect to the similarity measures that may be implemented; there is virtually no assumption on properties of that measure. Also, the speed of modern computers still permits this technique being applied at least for medium-sized case bases.

10.1.2 Indexing Techniques for Case Retrieval

Techniques employing explicit indexing (Barletta and Mark 1988; Rissland, Skalak, and Friedman 1993) allow for the integration of domain-specific knowledge, for example by specifying important or relevant features. However, these techniques assume that a set of appropriate indices may be chosen *a priori*, i.e. before the query is known. Hence, they provide only limited flexibility. Also, (Waltz 1989b) and (Thagard and Holyoak 1989) provide a number of arguments why this type of indexing is not appropriate from a cognitive point of view.

Case Retrieval Nets, too, can be considered as a structure indexing the case memory. However, a crucial difference is that these indexes can be directly obtained from the case representations: *Each IE comprises an index.*

10.1.3 Relational Retrieval

In particular for the integration of CBR systems with databases (Kitano, Shibata, and Shimazu 1993; Kitano, Shibata, Shimazu, Kajihara, and Sato 1992), relational retrieval (Wess 1995, pp. 165) has been designed. Given a query, a set of possible alternatives for each feature value is computed and a suitable (SQL) query is posed to the database system.

A principle problem with these techniques is that they are always implemented *on top* of a traditional (e.g., relational) database. Thus, they inherit major disadvantages, such as a set-based representation of the results rather than a preference ordering as desired for Case-Based Reasoning. To obtain this, additional effort is required on the CBR side to evaluate the result delivered by the database.

Also, as we have seen in Section 5.3.4, two cases may still be similar to a certain extend despite they being highly dissimilar with respect to a single feature. Consequently, to guarantee completeness and correctness of relational retrieval, the query posed to the database would have to include virtually *any* alternative value for all the features — thus posing a severe efficiency problem. Otherwise, if (similarly to indexing approaches) the set of possible alternatives is limited or specified *a priori*, cases may not be retrieved despite a higher degree of similarity than those delivered by the database system.

10.1.4 Hierarchical Memory Structures

Hierarchical approaches, such as *discrimination networks* or *shared feature networks* (Kolodner 1993, pp. 295) are used to reduce the search space compared to linear search by storing the case memory in a tree-oriented memory structure.

While tree-oriented search can be implemented highly efficiently, such techniques show a number of shortcomings with respect to flexibility of case retrieval:

- Firstly, this assumes that the case memory can be partitioned with sufficient quality independently of the expected queries.
- Secondly, the tree structure pre-determines an order in which components of the case are asked for and which causes in particular problems with *missing values*.
- Thirdly, a specially marked *target* feature, or class, is often required for building such a structure.

A further disadvantage is that building these memory structures is often computationally expensive, for example when certain optimality criteria should be satisfied. This is disadvantageous in particular when the case data will be updated often, as is the case for the VIRTUAL TRAVEL AGENCY described in Chapter 8.

10.1.5 *k*d-trees

Friedman, Bentley, and Finkel (1977) developed the technique of *k*d-trees for accessing multi-dimensional data in database systems efficiently. From an AI point of view, these structures are similar to decision trees known from Machine Learning (Quinlan 1990; Quinlan 1993).

Within the INRECA[1] project (Althoff, Auriol, Bergmann, Breen, Dittrich, Johnston, Manago, Traphöner, and Wess 1995; Auriol, Wess, Manago, Althoff, and Traphöner 1995), a range of possible combinations of inductive methods and CBR have been investigated. One result of this project was an extension of *k*d-trees such that the concept of similarity can be incorporated such that these trees can also be used for case retrieval (Wess, Althoff, and Derwand 1993; Wess 1995).

The idea here is

- to build a *k*d-tree for a given case base which results in a clustering of the case memory in multi-dimensional space;
- to utilize this tree as a decision tree for finding a similar case to a given query;
- to use specific tests, called BWB[2] and BOB[3] in order to verify whether neighboring clusters may potentially contain cases more similar than the ones found so far (Wess 1995, Chapter 8)

In a certain sense, *k*d-trees can be considered as *decision trees with backtracking* which makes it possible to guarantee completeness and correctness of case retrieval. Also, some of the typical disadvantages of hierarchical methods, such as problems with *missing values*, can be avoided (on the expense of performance).

On the other hand, building and updating the required memory structures is computationally expensive. Also, *k*d-trees can only be used if cases are being represented by means of feature values. It seems impossible, for example, to use this memory model for the SIMATIC KNOWLEDGE MANAGER and Textual CBR in general.

In how far other types of tree–like data structures, such as those used within the database community, provide better means for the retrieval in a CBR system (in terms of efficiency, flexibility, or both) is still an open issue.

10.1.6 FISH & SHRINK

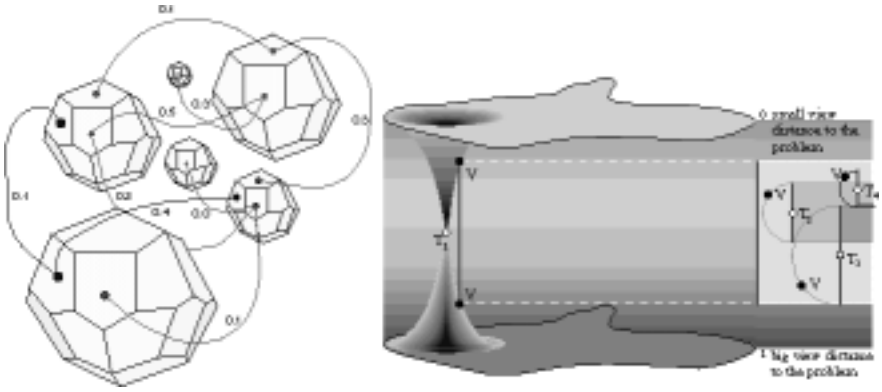
FISH & SHRINK is a technique developed within the FABEL project which addressed architectural design of industrial buildings (Gebhardt, Voß, Gräther, and Schmidt-Belz 1997; Börner 1998). Similarly to Case Retrieval Nets, FISH & SHRINK also employs a case memory structured as a net (Schaaf 1995; Schaaf 1996; Schaaf 1998). More precisely, so-called *aspects* are used to connect cases with each other that are similar in that aspect.

The fundamental idea underlying the FISH & SHRINK approach is that, for any given query q, the case base can be grouped into cases with a similar degree of similarity. More precisely, if two cases c_1 and c_2 stored in case memory are highly similar to each other and c_1 is similar to q, then c_2 will also be similar to q; if, on the other hand, c_1 is very dissimilar to q, then c_2

[1] Induction and Resoning from Cases
[2] Ball Within Bounds
[3] Ball Overlaps Bounds

can hardly be similar to q. Hence, once the degree of similarity between c_1 nd q are known, one can determine a *similarity interval* for c_2 and q, that is a possible range of similarity. These relationships are a direct consequence of the trial inequality in metric spaces. Consequently, if cases are being positioned as objects in some space, then computing the similarity of c_1 to q will not only result in an exact similarity value for these two but also in neighboring cases being *dragged down* as illustrated in Figure 10.1(b).

To utilize this property for case retrieval, the case memory for FISH & SHRINK is a set of multiply linked set of polyhedrons each of which represents a case. Each polyhedron consists of several aspect representations linked to corresponding aspect representations of other cases (*neighbors*) by edges labeled with the calculated distance between the two cases concerning the corresponding aspect (cf. Figure 10.1(a)).



(a) (b)

Figure 10.1: Illustration of Fish & Shrink: (a) The case base as a network of cases. (b) The retrieval metaphor of sinking cases being dragged down by others (from (Börner 1998))

Given this, retrieval is performed by

1. initializing the similarity interval for all cases
2. selecting (FISHing) particular case
3. comparing this case in depth to the query (thus SHRINKing its similarity interval to a single value)
4. adjusting (SHRINKing) the similarity intervals of all its neighbors according to the triangle inequality

This process is continued until

(a) all similarity intervals have been SHRINKed to a single value and, thus, the exact degree of similarity of each case to the query has been computed

(b) a sufficiently similar case has been found and it can be guaranteed that no other case will be more similar (according to the similarity intervals)

(c) the process is interrupted by the user

FISH & SHRINK in the above process makes use of the fact that computing the actual similarity between cases may be computationally expensive whereas assessing the similarity to neighboring

cases (via so-called *aspect distances*) requires much less effort. This has been a specific property of the FABEL domain.

A major advantage of FISH & SHRINK is that it allows for an *any time* retrieval: Whenever the retrieval process is aborted, precise statements can be derived about the similarity intervals computed so far, i.e. about cases that might be similar to the given query and cases being definitely dissimilar.

While at first glance this may look highly similar to Case Retrieval Nets, a closer investigation shows major differences: Firstly, FISH & SHRINK — like the more traditional *top-down* search methods — excludes cases sufficiently dissimilar and thus performance retrieval by rejection. Hence, in the worst case all cases have somehow to be considered in order to find the best ones during retrieval.

Secondly, the net structure of FISH & SHRINK represents pre-computed similarities between *cases* while CRNs only utilize pre-defined similarities between IEs.

Thirdly, FISH & SHRINK requires the triangle inequality to hold for a given similarity measure. Schaaf (1998) also describes how this assumption can be weakened to some extend by estimating to what extend the triangle inequality is violated for a given case base and computing an *error rate* which can be considered when determining the similarity intervals. Case Retrieval Nets, in contrast, to not require such an assumption. According to our experiences from the performed projects, we even doubt whether formal properties of metric spaces can be assumed for real-world similarity measures. Very often, symmetry can not be presumed — not to speak of the triangle inequality.

As it turns out, FISH & SHRINK is particularly useful in domains such as the one used in the FABEL project, where structural similarities play a crucial role in similarity assessment whereas the surface properties of cases are only of minor importance. Cognitive scientists also coined the term *gestalten* for these kind of structures which are recognized by humans without actually considering all the details of such a situation (cf. also Schaaf 1994).

10.1.7 CRASH

The CRASH memory model (Brown 1993; Brown 1994) is perhaps the one showing most similarities to the Case Retrieval Net approach:

The case memory also consists of a net model the nodes of which describe components of particular cases and the arcs of which encode certain relationships between these nodes.

Retrieval is performed by

1. Activating the nodes in memory corresponding to the query (here: *seed nodes*)
2. Propagating activation through the network encoding domain (or world) knowledge (see below)
3. Propagating activation to nodes associated to stored cases

Or, as Brown (1994, p. 16) describes it:

> *"... assume a certain knowledge item is associated with a case as one of its features. In the representation, the case will be explicitly connected to all other cases possessing the same feature via the single node representation of that feature."*

While at first glance the CRASH approach seems highly similar to CRNs, both the memory model and the retrieval process differ significantly. The memory model differs as in CRASH an extensive *world knowledge* is required to map from the nodes corresponding to the query to those nodes representing stored cases while CRNs simply employ arcs representing similarity. This *world knowledge* is formalized by means of *templates*, that is frame–like structures describing events and associated actors, requisites, expectations etc. Also, the links connecting the nodes do not

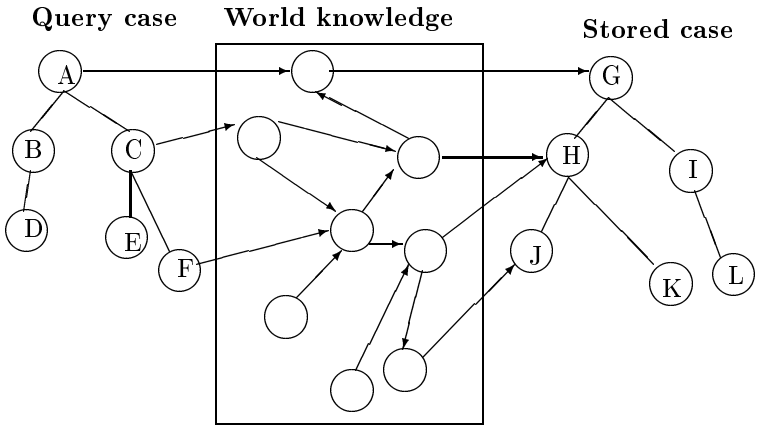


Figure 10.2: Sketch of the CRASH memory structure using *world knowledge* to relate a query to cases stored in case memory (adapted from Brown (1994))

directly represent the notion of similarity but rather the degree of *typicality* and *idiosynchracy*. These measures are based on a statistical evaluation of the stored cases and provide a means for determining the weights of various links when a query is posed to the system.

Due to this different memory model, the retrieval process differs, too. Instead of specifying additional relationships for the knowledge container of the similarity model, this is derived from the case base by statistical means and from a fairly general world knowledge.

Major shortcomings of CRASH are probably that the employed *world knowledge* has to be acquired somehow, and that it may grow rapidly for real-world applications. In fact, such a tendency can already be observed in the figures describing quite simple episodes, such as (Brown 1994, p.34f.). Hence the question of scalability arises — and the reported tests with case bases containing 8, 13, and 22 cases in a toy domain do *not* really support the claim of scalability. Compared to this, CRNs have been applied successfully to huge case bases as, for example, in the VIRTUAL TRAVEL AGENCY domain.

Nevertheless, a number of interesting ideas have been proposed, most notably the *leader share spoils model* — an approach to limit the spread of activation to the most promising nodes instead of propagating tiny amounts of activation to all nodes. This technique allows to focus on those nodes which become activated from a number of concepts expressed in the query and thus helps to suppress the investigation of nodes which came into consideration by pure coincidence.

According to private conversation, the CRASH system has also been designed for retrieving cases from different domains like in analogical reasoning instead of cases belonging to the same domain, as typically in CBR. This could not be provided with the Case Retrieval Net model.

10.2 Related Work in Textual Case-Based Reasoning and Information Extraction

As mentioned in Chapter 9 issues of knowledge management and the handling of textual information become more and more a focus of research in Case-Based Reasoning. This is primarily because these topics, in some sense, are related to utilizing experiences — a task that CBR claims to provide methods for.

According to the taxonomy of Textual CBR systems suggested by Burke (1998), the projects presented in Chapter 9 clearly belong to the group of *question answering systems* as they mainly work on textual data, consider this data to contain answers to the user's queries, and do not

perform any *deep reasoning* on the textual descriptions, such as summarizing or clustering texts, or assembling new documents. A considerable amount of reasoning is, in fact, performed *before* a Textual CBR system is being installed, namely when specifying the contents of the knowledge layers. If, for example, WordNet is utilized for this purpose, then the knowledge acquisition for the Textual CBR system can benefit from several dozen men years that went into the development of WordNet.

In the following, we will discuss other work in the area of Textual CBR.

10.2.1 FAQFinder

The FAQFinder project (Burke, Hammond, Kulyukin, Lytinen, Tomuro, and Schoenberg 1997a; Burke, Hammond, Kulyukin, Lytinen, Tomuro, and Schoenberg 1997b; Burke, Hammond, and Kozlovsky 1996) tries to apply CBR technology, in combination with other techniques, to document management. In particular, FAQFinders goal is to answer natural language questions by retrieving these from frequently asked questions (FAQ) files from USENET news groups FAQs. Hence, FAQFinder is a *question answering system*, too.

FAQFinder, like the CBR-Answers system for domains with English documents, uses a thesaurus (namely, WordNet, Miller 1995) to base its reasoning on a semantic knowledge base and assumes that documents are given in a semi-structured format (namely, as questions-answer (QA) pairs).

FAQFinder differs from the projects described in detail in Chapter 9 in so far as it does not focus on a specific domain. Instead, it applies a two stage process:

- In the first step, a shallow analysis, mainly of the keywords contained in the query, is used to infer the most likely news groups related to the request.
- After the user has decided on one of the presented news groups (i.e., after s/he selected a topic to focus on), a more sophisticated analysis of the related FAQ file starts to compare the contained QA pairs with the query entered by the user.

In some sense, this interactive scenario relates to the focus on a specific domain in that the user confirms the topic suggested by FAQFinder in the first stage. With the various CBR-Answers projects, we have focussed still further on specific domains in that each application has been customized specifically for a particular domain. For example, in technical areas, a lot of terms exist that would hardly be represented in general purpose thesauri, such as WordNet. Also, a careful knowledge engineering process has been undertaken to employ domain-specific knowledge for similarity assessment.

This would, in principle, be possible in the scenario of FAQFinder by encoding specific knowledge for each news group. However, this would obviously require a tremendous amount of knowledge engineering. To eliminate this, FAQFinder uses the same semantic base (i.e. WordNet) for all news group topics.

10.2.2 Spire

The Spire system introduced by Daniels and Rissland (1997) uses a completely different approach for dealing with textual cases. Based on the observation from the field of IR that people have problems in formulating *good queries* to IR systems, the idea behind Spire is to use a combination of CBR and IR technology:

- Given a user's request, a Hypo-style CBR module is used to analyze this request semantically and select a small number of relevant cases representing text documents.
- The most relevant cases from the first stage are then used to pose a query to the Inquery retrieval engine.

- After having retrieved relevant documents, cases are used again to extract the most relevant passages from the documents.

Compared to the projects described in Chapter 9, this is a completely different approach in which CBR is, in fact, used as an interface to IR, i.e. for constructing better queries to the IR system based on prior experiences about the system's usage (Daniels 1997; Daniels 1998). The use of specific domain knowledge is limited to the cases suggesting good indices for the IR system; it cannot be used for similarity assessment, etc.

10.2.3 Automatic Index Assignment

Brüninghaus and Ashley (1997) very much emphasize the *reasoning* aspect of Case-Based Reasoning, that is drawing inferences is crucial in their projects. Here, the objective is to have a set of more abstract feature descriptions and utilize Machine Learning techniques in order to automatically assign these indices to textual cases. Thus, a more structured representation of the cases can be obtained.

In particular, their work builds upon the CATO project (Aleven and Ashley 1996), a CBR system in the domain of legal argument. More specifically, CATO uses so-called *factors*; a factor being a kind of fact pattern which describes a certain legal situation on a more abstract level.

Again, this is a completely different approach to the systems presented in Chapter 9. In particular, CATO assumes that a carefully selected set of factors (or even a factor hierarchy) is available. Then, a real inference process can be performed rather than a pure retrieval only as in the CBR-ANSWERS system.

Nevertheless, ideas from this project might become useful if the functionality currently provided by the systems presented in Chapter 9 has to be extended in such a way that aspects of text classification become useful. For example, one could imagine that FAQs are classified according to whether they contain problem descriptions, requests for extended functionality, questions about available products, and so on. This information surely could provide further information and thus improve the performance of the systems.

10.2.4 Information Extraction for Document Analysis

As discussed in Chapter 9, Information Extraction techniques are used within the CBR-ANSWERS system in order to identify pieces of structured information in the running text. Currently, this is limited to identifying feature values for a predefined set of attributes.

> **Example 10.1** In the SIMATIC KNOWLEDGE MANAGER, for instance, the type of a `CPU` is often relevant. Hence, we utilize Information Extraction techniques for identifying CPU numbers from expressions such as `CPU 944`, `CPU-944`, or `944-series CPU`.

In terms of the tasks that can be addressed by Information Extraction (Cunningham 1997; Riloff and Lehnert 1994), this refers to the *Named Entity Recognition* subtask only. The more elaborated subtasks of Information Extraction are currently not implemented but research is being performed which aims at obtaining more structured representations of the textual descriptions by means of more advanced Information Extraction techniques (Glintschert 1998). More precisely, attempts are being made to use special sentence patterns both for representing the semantics of sentences as well as for classifying texts according to their contents. These patterns will contain the most relevant information extracted from particular phrases in a similar way as traditional AI frames and scripts. Also, these patterns can be learned automatically to some extend as described by Soderland (1997).

10.2.5 Information Extraction for Knowledge Acquisition

Apart from using Information Extraction for analyzing the contents of documents, related techniques may also be used during the knowledge acquisition phase. For example, Cardie (1993) describes a case-based approach for constructing the concept dictionaries that are required for filling the *knowledge layers* presented in Section 9.2.3. This and related approaches (Soderland, Fisher, Aseltine, and Lehnert 1995; Riloff 1991) are based on analyzing a sample document collection and extracting re-occurring expressions according to some predefined patterns.

To a very limited extend, such techniques have already been employed in the SIMATIC KNOWLEDGE MANAGER project in that, for example, the names, numbers, and versions of various devices have been extracted from documents (such as in the `CPU` example above) and the domain experts were asked about the relationships between the various types of devices.

In addition to these more high level methods, simpler heuristics might also be helpful for extracting phrases from a document collection that appear to be useful for representing the contents of the documents. The INFOFINDER project, for example, utilizes heuristics about formatted texts, capitalization etc. to automatically extract such phrases (Krulwich and Burkey 1997). Thus, the burden of knowledge acquisition required for following the Textual CBR approach presented in Section 9.2 could be avoided to some extent. However, such techniques are definitely not sufficient as they do not provide information about how the extracted phrases relate to each other. Furthermore, the heuristics listed by (Krulwich and Burkey 1997) are not directly applicable to German texts; for example, all nouns will be written in capital letters.

10.3 Related Work in Knowledge Management

As already discussed in Section 9.1, knowledge management (KM) is concerned with all processes related to discovering, collecting, preserving, and disseminating knowledge within some company or organization (O'Leary 1998a; Abecker, Bernardi, Hinkelmann, Kühn, and Sintek 1998). This also includes tasks, such as workflow management or the specification of an appropriate infrastructure for KM, which are beyond the scope of this thesis.

10.3.1 Organizational Memories

One of the major areas within KM is concerned with building organizational memories where the term

> *"... organizational memory refers to stored information from an organization's history that can be brought to bear on present decisions..."* (Walsh and Ungson 1991).

Given that definition, the close relationships between KM and CBR become obvious. In a limited sense, the Textual CBR approaches presented in Chapter 9, can be seen as a building block for an organizational memory in that they provide means for disseminating and reusing the knowledge contained in documents.

Generally speaking, many researchers in KM see CBR as a tool for implementing at least certain modules of knowledge management systems, be it that they explicitly mention it (Stanoevska-Slabeva, Hombrecher, Handschuh, and Schmid 1998) or that they describe processes which directly suggest the use of CBR tools (Röpnack, Schindler, and Schwan 1998).

An interesting aspect, however, that comes into play with organizational memories in KM is that such a system should play an active role in that it observes the user doing his/her work, anticipates potential problems, and takes the necessary actions. For this, very often techniques from research on intelligent agents are being used (Mahe and Rieu 1998). In contrast, the approaches presented so far in CBR are more or less passive in that a user explicitly has to query the system.

10.3.2 The Use of Ontologies

A further branch of knowledge management is concerned with the construction and utilization of ontologies in which the knowledge about an organization can be encoded formally such that an inference process can be performed (O'Leary 1998b; Benjamins, Fensel, and Perez 1998; Fensel, Erdmann, and Studer 1998). A major problem with such ontologies is that they require a considerable amount of knowledge engineering for being built and maintained. Thus, the question arises who is willing to invest in these processes and whether it is really worth the effort:

> *"... unless there is an overwhelming benefit that justifies the effort of constructing an ontology, it is unlikely that any common user would contribute to that activity..."* (Nakata, Voss, Juhnke, and Kreifelts 1998).

In terms of knowledge management vocabulary, the knowledge layers presented in Section 9.2.3 can be considered as *weak ontologies* in that these layers contain the knowledge which would go into an ontology in a typical KM system. However, knowledge layers have been designed much more from a pragmatic point of view in that they are far from being complete and correct — whatever that might mean in a natural language environment. Consequently, these knowledge layers do not directly provide an environment for performing a real inference process.

10.3.3 Textual Knowledge Management

As in Case-Based Reasoning, a sub-area of knowledge management is also concerned with the utilization of knowledge contained in textual documents. In contrast to the approaches presented in Chapter 9, however, the primary goal is not to find relevant documents but to extract concepts and their relationships from an existing document collection (Feldman, Fresko, Hirsh, Aumann, Liphstat, Schler, and Raiman 1998). For this, either techniques from collaborative filtering (Nakata, Voss, Juhnke, and Kreifelts 1998) or from intelligent agents (Ballim and Karacapilidis 1998) are most often used.

These techniques might turn out to be useful for the knowledge acquisition required for a Textual CBR system as explained in Section 9.2.3. Work in that direction is currently being carried out.

10.4 Related Work in Cognitive Psychology

As retrieval of cases in technical systems has the counterpart of remindings in human intelligence, there are, of course, a number of approaches worth to be mentioned originating in the cognitive science community. These include theories about the human brain in general, techniques for language comprehension, and approaches for (analogical) reasoning. As all these are well-established areas of research which brought forth a wide variety of highly complex theories and techniques, we will focus here strictly on the relationships to the task of case retrieval and a comparison to the Case Retrieval Net models.

10.4.1 Parallel Distributed Processing

In a certain sense, Case Retrieval Nets are closely related to the models developed in the *Parallel Distributed Processing* (PDP) group of Rumelhart, McClelland, and the PDP Research Group (1986). In particular, both approaches rely on using a distributed representation of the data which is accessed via some net–like structures (cf. the figure on page 28 of (Rumelhart, McClelland, and the PDP Research Group 1986)).

However, there are major differences concerning the underlying theoretical framework, the semantics associated to the different models, and the application areas:

- The units in the PDP models typically do not have any meaning when considered in isolation. They are in some sense similar to hidden nodes in Artificial Neural Networks (ANNs). Meaning strictly arises from *patterns of activations*. In CRNs, on the contrary, the units represent atomic parts of cases which we called information entities. Hence, every such unit contains a piece of knowledge and is itself meaningful. This allows for the interpretation of processes going on in these nets and thus provides a means of explanation.

  Because of this, CRNs would even be classified as having a *local* rather than a *distributed* representation in terms of the PDP model. We think, however, that the fact that IEs are represented just once and *shared* by their associated cases justifies the claim of a distributed representation in CRNs.

- Similarly to ANNs, the connections between units in the PDP models represent how strong one unit supports another. Again, there is no meaning in the real world directly associated whereas in CRNs connections express the concept of similarity and relevance. Consequently, learning is CRNs is applicable to a lesser extend than in PDP models whereas knowledge about the domain can be integrated more easily. In how far learning can be used *after* a CRN has been constructed, for example for *fine–tuning* the similarity and relevance relationships, is still an open question.

- Due to the distributed representation, learning (training) is a necessary prerequisite in PDP models in order to find the representation itself and to adjust the weights. In this sense, the knowledge contained in prior examples is *compiled* to obtain the weights. In CRNs, on the contrary, these cases are directly utilized for solving problems and the net structures are only used to access the most relevant ones. In particular, case nodes corresponding to cases are explicitly present in this model.

  Learning may, however, be used to fine-tune some memory structures of CRNs (cf. Section 6.2).

10.4.2 Marker Passing Algorithms

Marker passing algorithms (Charniak 1983; Wolverton 1995) rely on a semantic network the nodes of which represent single concepts. Inference within these networks is performed by passing special annotations (*markers*) from specific nodes to their neighbors. This process can be performed in a highly parallel manner if, for example, for each node in the network a single processor is used. By investigating all the nodes to which a particular marker was passed, paths in the network can be found and/or investigated and, thus, inference can be performed. Such marker passing algorithms already played a major role in the development of the CRASH system discussed in Section 10.1.7.

As an example consider Figure 10.3: The semantic network encodes knowledge about animals. If the goal is to find out whether `Daisy` is `warm blooded`, two distinct markers are placed on the two nodes representing these two concepts. Then, these markers are passed on to all the neighboring nodes (possibly along arcs of selected types only). Finally, at the node representing the concept `mammal` these two markers *meet*, meaning that `Daisy` is a `mammal` and that `mammals` are `warm blooded` — hence, the question can be answered positively.

Major disadvantages of marker passing algorithms are that

(a) the *search* in the network is rather undirected: in some empirical studies about 90% of all the investigated paths yield a dead end because the spreading activation process essentially performs a blind search (cf. Wolverton (1995) for a discussion);

(b) the decision of whether to pass a marker is strictly a *yes-or-no-decision*: there is no means of expressing a certain degree of membership to a concept.

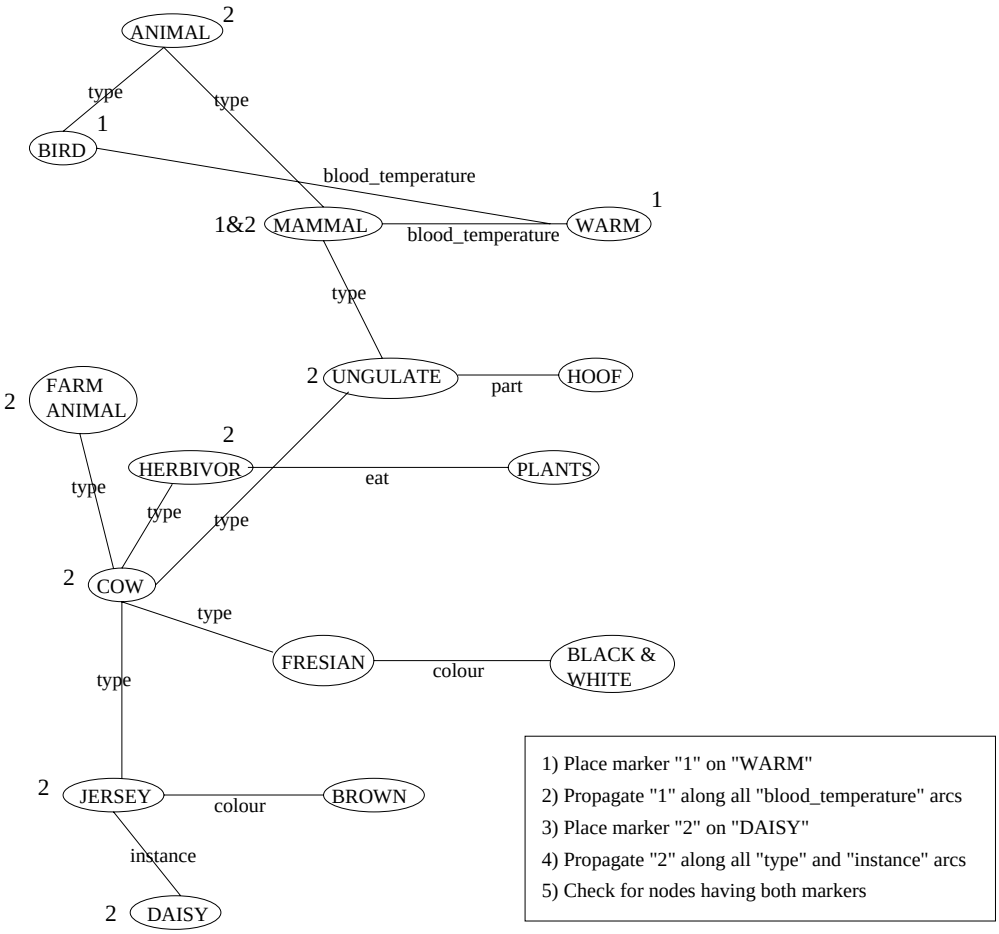


Figure 10.3: Illustration of Marker Passing algorithms (adapted from Brown (1994))

Concerning both weaknesses Case Retrieval Nets are superior:

ad (a) The search space is reduced as the number of propagation steps is limited according to the formal definitions given for the activation process in Section 5.2. In most applications, the spread of activation is further reduced because only a limited number of non-zero similarity arcs exists. When a composite similarity measure is used, for example, it does not make sense to compare two values of completely different attributes.

ad (b) Instead of a binary *yes-or-no-decision*, the degree of similarity is expressed by the activation value passed along the arcs in the net.

10.4.3 Spreading Activation Theories: ACT*

Very briefly, the main points of the ACT* learning theory by Anderson (1988) are twofold: Firstly, the architecture of the reasoning system consists of three different memories: a declarative, a procedural, and the working memory. Secondly, the inference process heavily relys on a spreading activation process in the working memory, that is:

> *"...memories can be brought back into working memory by a spreading activation process by which associated concepts retrieve ...facts"* (Anderson 1989).

where a *fact* corresponds to some kind of concept.

This differs from the above described marker passing algorithms in several aspects, first of all in values being propagated through the net rather than boolean markers. In terms of the ACT* model, a distinction between the various types of knowledge can be observed in Case Retrieval Nets, too:

The nodes of the net containing IEs and cases ($E \cup C$) make up the *declarative* part, while the knowledge encoded in the similarity and propagation functions (σ and π_n) describe *how* the overall net structure will change depending on activation. This could be considered in some sense as *procedural* knowledge. The *working memory*, as cited above, is the part of the CRN currently being active, i.e. sufficiently activated.

While in ACT* there is just one set of concepts (i.e. nodes in the net), which are used for starting the spreading activation process, propagating the activation through the net and retrieving knowledge, in CRNs two sets E and C are distinguished — the first one being used for starting and performing the propagation process, the second one containing those memory items that are searched for. Of course, the formal framework allows that $E = C$ — yielding the memory structure of ACT*. Since our main goal is retrieval of cases, however, it seems reasonable to distinguish these sets.

10.4.4 Knowledge-directed Spreading Activation

An improved version of the basic spreading activation algorithms which has been developed in particular for retrieval purposes in analogical reasoning, is *Knowledge-Directed Spreading Activation* (KDSA), as described by Wolverton and Hayes-Roth (1994) and Wolverton (1995).

In KDSA, the spreading activation process is performed in several steps: Each time an analogue has been retrieved from memory, the quality of it is evaluated by a heuristic mapping component. Based on the degree of usefulness determined by this mapping component, some

> *"... search control module modifies the direction of subsequent spreads of activation into more promising areas of the knowledge base"* (Wolverton 1995).

As expected, theoretical investigations as well as empirical tests proved that KDSA is superior to pure spreading activation.

As pointed out, however, KDSA has been designed mainly for cross-domain analogies where the goal is to retrieve *"... semantically distant analogies"*. This obviously differs significantly from out viewpoint of CBR: While we, too, demand a high flexibility in terms of memory representation and retrieval, we assume that the goal of retrieval is to access previously encountered cases describing problem situations in the *same* domain. Given this assumption, the task of the heuristic mapping component of KDSA (namely to determine how close a retrieved case is to the query) can be fulfilled by simply assessing the similarity between the query and the retrieved case — that's exactly what Case Retrieval Nets do.

Furthermore, if retrieval is not performed as defined in the formal model of BCRNs (Definition 5.3) but by *Lazy Propagation of Similarity* as described in Section 6.4), then retrieval in CRNs becomes similar to KDSA in so far as spread of activation is limited and hence blind search is avoided. In contrast to the *heuristic* KDSA approach, however, one can proof formally that retrieval in Case Retrieval Nets directly implements a given similarity function and, hence, no retrieval error will occur.

10.4.5 ROBIN / REMIND

ROBIN/REMIND is a system for retrieval of episodes utilizing a structured spreading activation network (Lange and Wharton 1993; Lange 1995). In order to be able to infer where a particular activation comes from, unique *signatures* are used to solve the *variable binding problem*. This is only partially possible in the Case Retrieval Net models: In principle, the propagation functions

within case and IE nodes could be extended such that they keep track of where a certain amount of activation came from. In practice, however, this would cause the model to lose much of its simplicity and efficiency.

In ROBIN/REMIND no direct similarity relations exist; instead, a fairly complex world knowledge is used to determine related concepts and the amount of activation to be passed. Even with just a small number of short episodes (i.e. single sentences) highly complex networks with a number of different *signatures* emerge. Also, in the reported experiments it is not quite clear whether the minor differences in activation for the various concept nodes are really due to the inference process or whether the initially specified weights are responsible for these.

Currently, this direction of research has not been addressed in Case Retrieval Nets. If it really matters *why* a particular case has been activated, we suggest to perform a retrieval process as usual and to compare the retrieved case(s) piece by piece with the query by means of some other technique, such as explanation-based techniques (Branting 1991; Bergmann, Pews, and Wilke 1993; Bareiss, Branting, and Porter 1988; Kass 1986; Leake 1990). Due to the amount of data that had to be dealt with in the presented applications, the case memory and the retrieval process should not be overloaded with any additional structure not really required for retrieval.

10.4.6 Analog Retrieval

To find a similar situation in analogical reasoning, some structural relationships are searched for which can be observed in both, the current problem situation and a stored past situation (Gentner 1983). Very often, this kind of analogy is applied across domains.

To apply the Case Retrieval Net framework for the retrieval task in CBR we do not focus so much on *structural* similarity but on *surface* similarity: As usually no domain change occurs, similar problems are assumed to be represented by similar descriptions, such as a list of attribute-value pairs having similar values.

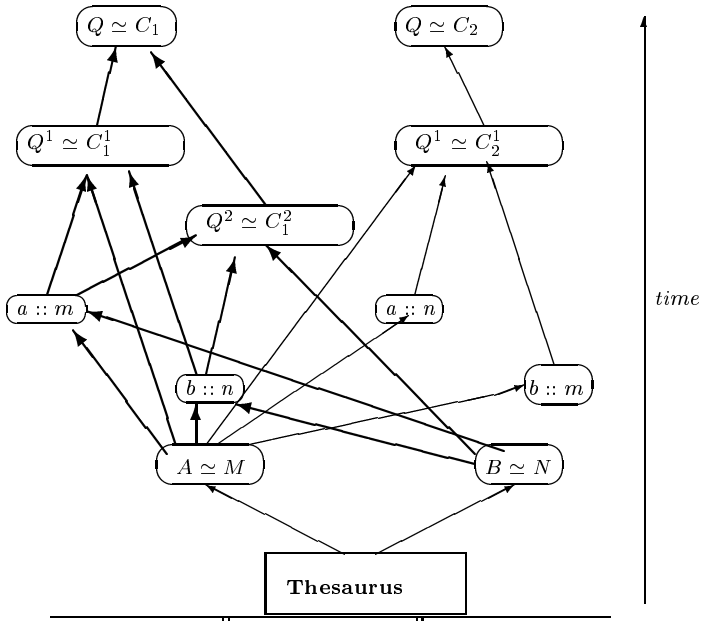
An interesting approach to analog retrieval, which also relies on a net structure representing the problem situations, has been described by Thagard, Holyoak, Nelson, and Gochfeld (1990): The ARCS system utilizes some background knowledge about similar relations in order to subsequently construct a net useful for accessing situations in memory which have several kinds of correspondences with the structure of the problem situation.

In particular, *semantic similarity*, *isomorphism*, and *pragmatic relevance* play a major role in the ARCS system. While the first directly corresponds to the similarity function (expressed by σ) in CRNs, the latter support goal-dependent analogies and are in some sense related to special contexts expressing particular weights of certain features. Isomorphisms do not have a direct counterpart in CRNs.

Given a problem, or *probe*, a constraint network is built by using knowledge about similar predicates contained in a kind of thesaurus. Via additional memory structures all stored analogs can be accessed which contained these predicates (cf. Figure 10.4). The constructed network also includes excitatory links according to the strength of similarity and inhibitory links between units representing incompatible parts in the network (e.g. different instantiations of the same variable). After this network has been constructed, all nodes are updated in a synchronous manner until the network settles in a stable state.

Both techniques, ARCS and CRNs, use a *bottom-up* approach to avoid search in the traditional AI sense in order to access relevant situations stored in memory — or, as Thagard, Holyoak, Nelson, and Gochfeld (1990, p. 281) describe it:

> *"It is important to be clear about what* ARCS *is* not *doing: It does not compare the probe with every structure in memory, but considers only those that have semantically similar predicates. Nor does it do a complete match between the probe and the source analogs whose potential relevance is indicated by semantic similarity."*



Query Q		Store case C_1		Store case C_2	
Q^1	A(a,b)	C_1^1	M(m,n)	C_2^1	M(n,m)
Q^2	B(b,a)	C_1^2	N(n,m)	C_2^2	R(n,m)

Figure 10.4: Construction of a net for retrieval of analogs in ARCS (adapted from Thagard, Holyoak, Nelson, and Gochfeld (1990))

A major difference between ARCS and the *Case Retrieval Net* approach is that the net structures are dynamically constructed in the first system. However, this requires that indices from all predicates to the probes in which these predicates occurred be stored and managed. If these data structures have to be kept in memory anyway, it seems straightforward to encode the knowledge about similarity not in a separate thesaurus but as binary relations between predicates (or arcs in the net) — this already appears to be very similar to the initial ideas of Case Retrieval Nets.

10.5 Miscellaneous

10.5.1 CONSYDERR

CONSYDERR is an approach to combine reasoning at two different levels (Sun 1995b; Sun 1995a): On the upper, the *conceptual level*, concepts of the domain are encoded in a local representation and explicitly given rules are used for performing an inference process. On the lower, the *microfeature level*, microfeatures are used to further describe the concepts of the upper level. Here, similarity relationships are used for inferences. A formalism called *Fuzzy Evidential Logic* is then employed to merge the result of both inference processes.

While at first sight this may seem similar to the *Microfeature Networks* of Section 6.2, there are important differences:

- According to the description by Sun (1995a), both the rules on the upper and the similarity relations on the lower level express basically the same knowledge: If two concepts on the

upper level are connected through a specific rule, then their microfeature representation has to be similar — and vice versa. Hence, if two concepts are connected via a rule, the complete cross product of their microfeatures will be activated. In MFCRNs, on the contrary, microfeatures are employed to determine similarity between two IEs, there are no corresponding rules.

- In CONSYDERR it is not clear where the knowledge structures (both, rules and similarity relations) arise from. Also it is not clear which parts of the domain will be considered *concepts* and which *microfeatures* (in (Sun 1995a), for example, `cattle_country` appears as a concept while `woodland` is a microfeature).

  In Case Retrieval Nets, the descriptions of cases determine the granularity: Everything which is explicitly expressed as part of cases and queries will be represented as IEs (here: *concepts*), lower level knowledge items may be microfeatures in MFCRNs.

.

10.5.2 Vague information in databases

Within the database community the need for handling imprecise data and vague queries has been identified, too. Traditional database models and commercially available database management systems only support two-valued logic (e.g., relational algebra). This implies that during retrieval of data a binary decision has to be made whether a data record answers a query or not. In particular, query processing strategies to increase efficiency heavily rely on this feature.

In contrast to these traditional database models, attempts have been made to include reasoning about uncertainty and similarity directly in database technology (Lee 1992; Motro 1988).

A probabilistic framework, for example, has been introduced by Fuhr (1990). In this approach, each query to a database system consists of a boolean part (as traditionally) and a vague part. While the former is answered by a set of *preselected objects* satisfying all criteria expressed in the boolean part of the query, the latter ranks the list of *preselected objects* according to the vague query conditions and some indexing weights given to the system.

Compared to retrieval in CBR systems, this approach shows a number of differences, most remarkably:

- Firstly, only a simplified notion of similarity is used. When ranking the *preselected objects*, the difference between numerical values can be determined and, thus, some kind of *distance* can be computed which can be taken into account during ranking. For symbolic values, however, a transformation to discrete intervals (such as *low*, *medium*, *high*) has to be done.

- Secondly, it is required that the query posed to the database explicitly expresses what is supposed to be part of the boolean query and what of the vague query.

- Thirdly, the indexing weights used during the ranking process are fixed and can not be changed from within the query. Hence, it will be difficult to express different priorities in varying contexts.

- Finally, the technique introduced is just an *add-on* to the existing database models: Basically it is an extended query language — the memory model itself remains unchanged. This has the advantage that one can still rely on the capabilities of the underlying database management system, such as efficient query processing, query optimization, recovery, or handling of parallel transactions.

10.5.3 Decision-theoretic approaches

Another area of research dealing also with the development of (so-called *normative*) expert systems is decision theory. Decision theory includes probability theory, where decisions are mainly based on the probability of certain events (e.g., diagnoses) given a set of known properties (e.g., symptoms).

Early results of this research have been *influence diagrams*, *knowledge maps*, and *belief networks* (Pearl 1988; Pearl 1986), which are based on the axioms of probability and decision theory (e.g., Bayes' theorem). An extension of this work has been developed by Heckerman (1991) who introduced the concepts of *similarity networks*:

- A *similarity network* consists of a number of *distinguished* nodes representing a set of mutually exclusive and exhaustive *hypotheses*.
- Edges in a *similarity network* indicate that a *local knowledge map* exists helping to distinguish between the two *hypothesis* connected.
- A *local knowledge map* for a pair of *hypotheses* represents dependencies among the *distinguished* node and a set of *non-distinguished* nodes helping to decide for either of the two *hypotheses*.
- Usually, *distinguished* nodes represent diagnoses while *non-distinguished* nodes represent observable symptoms.
- Inference in *similarity networks* is performed by assessing the set of *non-distinguished* nodes corresponding to the observed symptoms and using Bayes' theorem to compute the probability of the diagnoses represented by the *distinguished* nodes.

Although this gives only a rough overview over the idea of *similarity networks*, it already becomes obvious that — despite the name — these nets have little in common with the model of Case Retrieval Nets.

Firstly, we do not distinguish between hypotheses and symptoms during case retrieval. According to retrieval being considered as an information completion process, it is not even required that cases may be splitted into such separate parts.

Secondly, we do not want to explicitly compute the most probable diagnosis (or solution) to a problem but intend to retrieve the cases most *similar* to the given query. This implies that complete cases are retrieved allowing for a detailed re-interpretation by a human expert.

Thirdly, the arcs in the CRN models represent different degrees of similarity or relevancy. They do not correspond to any kind of probability.

The latter point is more crucial than might appear at first glance: It is highly questionable whether similarity of cases can, in general, be expressed by a kind of probability using, for example, Bayes' theorem for inference.

Example 10.2 For illustration, consider the following *events* from the VIRTUAL TRAVEL AGENCY domain: Let

$A =$ "customer travels to `Greece`"

$B =$ "customer travels to `Turkey`"

$C =$ "customer wants to go `bathing`"

Then, $p(A|C)$ and $p(B|C)$ refer to the probability that the customer travels to `Greece` or `Turkey`, respectively, given that s/he wants to go `bathing`. According to Bayes' theorem we could compute:

$$p(A|C) = \frac{p(C|A) \cdot p(A)}{p(C)}$$
$$p(B|C) = \frac{p(C|B) \cdot p(B)}{p(C)}$$

Assuming that both, $p(C|A)$ and $p(C|B)$ are nearly equivalent (most people traveling to either of the two destinations want to spend a `bathing` holiday), a difference between $p(A|C)$ and $p(B, C)$ is only due to the difference between $p(A)$ and $p(B)$, i.e. the destination is selected that has most often been selected before. But why should a customer go to `Greece` just because lots of people did so before?

Of course, this approach is reasonable in diagnostic applications: If a certain malfunction often occurred with certain symptoms, this increases the probability of that malfunction if the same set of symptoms is observed again.

While the above example refers to the probability of events belonging to different types of attributes (`destination` versus `leisure time facilities`), one might also think about describing similarities among values belonging to the same type of attribute using probabilities.

Example 10.3 One might be interested in the similarity between `Greece` and `Turkey`, expressible as $p(B|A)$: If `Greece` is desired, how likely is it that `Turkey` will be accepted? Again using Bayes' theorem this yields

$$p(B|A) = \frac{p(A|B) \cdot p(B)}{p(A)}$$

which contains $p(A|B)$ — which we do not know either.

Finally, we think that there is a principle drawback in using probabilistic methods for similarity assessment: As discussed in Section 2.2.5, the case base and the similarity measure are two distinct knowledge containers. If, however, the probabilities of certain events are obtained based on how frequent these events could be observed in the case base, then the similarity measure is derived from the case base and, thus, represents only knowledge that has been implicitly present in the case base before. Consequently, an essential part of the CBR system is missing, namely a similarity measure that captures the relationships between the various objects in the domain independently of the particular case base being under consideration.

An alternative approach, in which Case-Based Reasoning and decision theory are seen as complementary techniques, has been suggested by Tsatsoulis, Cheng, and Wei (1997). However, they

> *"... view CBR as a technology for automated, intelligent problem solving ..."*

and suggest to utilize decision theoretic approaches when dealing with uncertainty during the Retrieve phase of the CBR cycle. In a nutshell, the authors analyze the result of —what they

call— a standard retrieval phase and, thus, determine the values of previously unknown features as they occur in the retrieved cases. This information then allows for a more refined selection of the most similar case(s).

For the work presented in this thesis, this kind of approach is of limited interest only. Firstly, even Tsatsoulis, Cheng, and Wei (1997) confess that

> *"CBR systems can retrieve similar cases even in instances of uncertainty."*

By integrating another formalism, however, other problems might be harder to solve, such as whether the retrieval process is correct: As correctness and completeness have to be considered with respect to a given similarity measure, it will be difficult to even define correctness of the retrieval process if some part of the knowledge being used for retrieval purposes is based on some completely different formalism and the knowledge required for this is not accessible to the CBR system.

Secondly, as emphasized in Section 1.1.2, our focus is set on decision support systems rather than fully automatic problem solvers. Hence, in case of uncertainties or possible alternatives we would prefer consulting the user.

Chapter 11

Summary and Outlook

The growing amount of ongoing CBR research – within an AI community that has learned from its previous experiences – has the potential of leading to significant breakthroughs of AI methods and applications.
Agnar Aamodt, Tutorial at EWCBR, 1993

In this final chapter, we will summarize major results of this thesis as they have been worked out in the preceding chapters. Also, we will give an outlook on future work and open issues.

11.1 Conclusions

Having close relationships to both cognitive science as well as engineering disciplines, research in Artificial Intelligence can always be seen from three different perspectives:

The cognitive perspective is concerned with the attempts of constructing (psychological) models in order to understand human thinking and behavior.

The theoretic perspective is related to the development of formal methods for both the description of cognitive models as well as the specification of related systems.

The pragmatic perspective is concerned with the implementation of running systems that show some kind of intelligent behavior.

In this thesis, we have addressed primarily the latter two aspects. We have presented the formal model of Case Retrieval Nets that formed the basis for all the extensions and applications discussed thereafter. Furthermore, we have investigated the formal properties of this model, its advantages, limitations, and potential extensions.

In addition, we have presented a notion of problem solving called *information completion* that more accurately describes the processes required in a decision support environment. In particular, the *a priori* distinction between the problem description of a case and its solution is no longer required. Rather, the notion of *Information Entities* used to describe both cases and queries is a very central one. We have shown that more traditional approaches, such as classification, can be considered as special cases of information completion.

11.1.1 Advantages

As we have discussed already in the corresponding chapters, Case Retrieval Nets show a number of advantages:

- CRNs directly implement a given similarity measure in that they utilize a specified similarity function and a given case base for building a case memory that, during retrieval, behaves much like an associative memory. As a consequence, the question of correctness and completeness does not arise, i.e. CRNs can formally be shown to not make a retrieval error.
- CRNs are highly efficient which makes it possible to deal with case bases larger than used usually in CBR systems. The main reasons for this are that
  1. major parts of the similarity measure are *precompiled* in a CRN thus reducing the effort at retrieval time;
  2. the distributed representation of the case memory allows for a spreading activation process in which intermediary results can be reused rather than being computed again and again.

  This is in particularly true if a composite similarity measure can be used for a particular application.
- CRNs provide enough flexibility to implement decision support systems based on information completion processes. In particular, they do not require cases to have a specific element such as a *class* or a *solution*. Rather, CRNs should be used by human users in the sense of an external memory.
- The model of Basic Case Retrieval Nets can be extended in various ways in order to incorporate domain-specific knowledge or to implement different spreading activation strategies. In this thesis we have presented some possible extensions while others, for example stochastic approaches or learning techniques of artificial neural networks, have not been addressed at all.

An interesting interpretation of data and cases that already had been implicitly present in the other applications became particularly obvious in the area of Textual CBR, namely to distinguish between the existing data, the cases representing this data, and the indexes used for retrieving cases. For most applications in this thesis, we have chosen to consider cases as a *view* on data rather than constructing a case base that exists besides the actual data. This significantly differs from other models and, in particular, from most available commercial CBR products.

11.1.2 Limitations

As the name already indicates, Case Retrieval Nets have been designed to be used during the Retrieve phase of the CBR process. In that sense, it is *not* a stand-alone CBR system in that it does not cover aspects such as how the retrieved cases can be reused, what is required for adaptation purposes, or how learning could be integrated into the system.

Even for retrieval purposes, we have seen that there are limitations in so far as, for example, adaptation knowledge cannot be considered during retrieval. Also, Case Retrieval Nets might not be the best choice in domains where structural similarity plays an important role even though in many situations an equivalent *flat* similarity measure can be constructed.

Case Retrieval Nets as described in this thesis can definitely not be applied in domains where the internal structure of cases is crucial. This should be obvious already from the fact that we

always considered cases to *sets* of Information Entities, thus deliberately ignoring relationships among IEs that might potentially exist. As with structural similarities above, it might be possible in certain situations to come up with an alternative *flat* representation of cases that still captures the structural relationships, for example by defining additional IEs for describing the *secondary features*. In general, however, this will not be possible.

As mentioned in the beginning of this chapter, we have focussed primarily on CRNs as a model for the development of intelligent systems. Some vague ideas exist that there might also be close relationships to cognitive models (cf. Burkhard 1995b) but this topic has not been addressed in this thesis.

11.1.3 Applications

We have presented a general framework describing the specification, implementation, and maintenance of information systems that are based on the Case Retrieval Net model. In particular, we have described the process and steps required for developing a related system. Some of these processes are known from standard software engineering but deserve special attention if a Case-Based Reasoning system should be built.

In Chapters 8 and 9 a number of applications have been discussed in more or less detail. While each of these had its specific characteristics, all show properties related to information completion.

Applications in E-Commerce

In the area of electronic commerce, for example, we have argued that Case-Based Reasoning, in general, can provide major contributions in the design of systems that support and assist the user in searching product catalogues. In contrast to, for example, database technology, the knowledge that a CBR system can utilize allows it to suggest alternatives and rank these with respect to an expected utility for the customer.

The major benefits of Case Retrieval Nets in that area are that they provide enough efficiency to come up with suggestions virtually instantly. Also, they are flexible enough to deal with different needs of customers from vague to highly specific requests. These have been crucial requirements, for example, in the development of the VIRTUAL TRAVEL AGENCY that has been presented as a highly successful application in Chapter 8.

Furthermore, as we have seen in the discussion of the CBR-SELLS system, a major advantage of CRNs in that area is that the case memory, i.e. the index used for retrieval purposes, can be separated from the case base. Thus, even in environments with limited resources, such as when searching a CD-ROM product catalogue on a standard PC, large amounts of data can be dealt with.

Applications Related to Knowledge Management

Another application area that we covered in detail is the content-oriented retrieval of what we called know how documents. These are assumed to mainly textual representations containing specific know how about a domain. Typical examples for such documents are the common *Frequently Asked Questions* collections.

We have seen that by means of *Textual CBR*, systems may be implemented which are superior to more traditional Information Retrieval methods in that they can incorporate knowledge about the domain. As such a system deals with natural language texts, this knowledge will, of course, relate the various terms that occur in these documents but may also contain attribute–value based information as well as a description of the structure of the entire domain.

Again, a major advantage of the Case Retrieval Net model has been that this model is highly flexible. In particular, we have shown that by means of this model a highly efficient case

memory can be built despite the fact that the cases do only have a very weak internal structure as compared to, say, feature vectors or object-oriented representations.

As already discussed in Section 10.3, we only considered a highly specific part of knowledge management, namely the dissemination and reuse of knowledge contained in specific types of documents. Knowledge Management in itself is a very broad area ranging from data warehouses to workflow management and the technical infrastructure required for a successful knowledge management strategy.

11.2 Outlook

Following the three perspectives of research in AI discussed in the beginning of this chapter, areas for future work could also be grouped according to whether they primarily address cognitive issues, the theoretical framework, or the implementation of practical systems. In each of these areas, a number of interesting problems remain unsolved so far.

11.2.1 Extensions of the Theoretical Framework

As discussed above, Case Retrieval Nets so far cannot be directly applied in domains where a structural similarity should be considered or where cases have a richer structure, such as in an object-oriented representation. Currently, work is carried out addressing the later point. The main motivation for this is that for a number of applications an object-oriented case representation is beneficial in particular for modeling and maintaining the entire application.

As we have seen in the context of *Conceptual CRNs*, an equivalent (flat) BCRN can often be constructed. But the question remains whether a CRN could also make use of the object-oriented case representation and if so which is the better choice: compilation to a BCRN or utilization of richer structures.

Furthermore, a number of interesting questions arose from some of the applications that we dealt with recently. These questions even challenge some of the most fundamental things about CBR as they can be found in every textbook. For example:

- What should be the result of the Retrieve process in a practical system? Is it a ranked list of cases or just references to the cases as it is the case in CRNs? While this may seem a minor difference from a theoretical point of view, it will have severe consequences with respect to the design of a related system. For example, the answer to the above question will imply whether the retrieval system will have to keep the entire case base in memory or whether some external service is required for getting access to the actual case data.

- Related to the previous point is what additional information should be returned by a retrieval method. In many practical situations, it does not suffice to just deliver cases (or references to cases). Rather, information about what parts of the query have been analyzed, what distinguishes the retrieved cases, what feature would be best suited for a refined query and the like is often important. On the one hand, delivering this information is not directly related to retrieval. On the other hand, for obtaining most of this information the same analysis of the posed query has to be performed as for the retrieval.

- What, exactly, is similarity that the result of the Retrieve process will be based upon? According to the common understanding, similarity is a measure than somehow ranks cases with respect to a given query. In Section 5.3.4 we have discussed that viewing similarities as inverse distances might not be appropriate. But may be we have to go even further in so far as the Retrieve phase may not always deliver a single ordering of cases. Rather, in some situations multiple dimensions have to be considered and it simply is not possible to encode the relationships in a single one-dimensional measure.

11.2.2 Extensions for Practical Applications

E-Commerce

A requirement in electronic commerce environments is that an intelligent system not only suggests appropriate products and reasonable alternatives but also that a kind of dialog can be implemented which provides some guidance to the user (Wilke, Bergmann, and Wess 1998). As CRNs focus on the retrieval aspect of the CBR cycle, this is obviously not directly possible with that model. It has to be investigated, however, what kind of information a CRN should deliver in order to allow an external component to come up with such a dialog.

Textual CBR and Knowledge Management

The main area of research that we are currently engaged in is related to Textual CBR and knowledge management as discussed in Chapter 9. Our main objective here is related to the above mentioned idea of being able to represent richer structures in the model of CRNs. More precisely, one of the limitations of the CBR-ANSWERS system so far is that the structure of textual descriptions is discarded and sections of text are only represented by means of sets of IEs.

As we have discussed, Natural Language Processing approaches (which could more accurately capture the structure of texts) do not seem feasible because of various reasons. An alternative idea is to use pattern-based methods in order to recognize stereotypical phrases. Such patterns have been introduced as *concept nodes* in the Information Extraction community (Riloff 1991) and first promising steps into that direction have already been carried out within the THEMESEARCH system developed by Glintschert (1998). The application of these techniques to Textual CBR implies some additional requirements (Lenz and Glitschert 1999) which will be addressed in the near future.

Also, further research is required with respect to how the CBR-ANSWERS system can be enriched in order to be useful for knowledge management beyond document retrieval. Questions that appear to be of particular importance are:

- How could such a system be used in a distributed environment, i.e. when several document collections have to be managed, for example for different groups of experts each having a *local* version of the system? Agent-oriented techniques might come into play here.
- How could the knowledge acquisition required for the implementation of a related system by simplified? What techniques are possible to (semi-) automatically recognize relevant concepts and their relationships? Again, this is currently a very active area of research (Hübner 1999).
- How can the concepts being used as well as there relationships be visualized for providing means of explanation as well as validation? An interesting approach to this has been developed within the KNOWLEDGE GARDEN system introduced by Crossley, Davies, McGrath, and Reiman-Greene (1998). However, as we primarily deal with documents, this approach is not directly applicable. Also, the manual effort must be kept to a minimum.

Part V

Appendices

Bibliography

AAAI (1988). *Proceedings of the 7th Annual National Conference on Artificial Intelligence AAAI-88*, St. Paul, Minneapolis, USA. AAAI: Morgan Kaufmann Publishers.

AAAI (1991). *Proceedings of the 13th Annual National Conference on Artificial Intelligence AAAI-93*. AAAI: Morgan Kaufmann Publishers.

AAAI (1994). *Proceedings of the 14th Annual National Conference on Artificial Intelligence AAAI-94*. AAAI: Morgan Kaufmann Publishers.

Aamodt, A. (1991). Problem solving and learning from experience. *Nordisk AI Magazin 6*(1), 19–25.

Aamodt, A. and E. Plaza (1994). Case-based reasoning: foundational issues, methodological variations, and system approaches. *AI Communications 7*(1), 39–59.

Aamodt, A. and M. Veloso (Eds.) (1995). *Case-Based Reasoning Research and Development, Proceedings ICCBR-95*, Lecture Notes in Artificial Intelligence, 1010. Springer Verlag.

Abecker, A., A. Bernardi, K. Hinkelmann, O. Kühn, and M. Sintek (1998). Toward a technology for organizational memories. *IEEE Intelligent Systems 13*(3), 40–48.

Abecker, A., S. Decker, and O. Kühn (1998). Organizational memory. *Informatik Spektrum 21*(4), 213–214.

Aleven, V. and K. D. Ashley (1996). How different is different? See Smith and Faltings (1996), pp. 1–15.

Allen, B. P. (1994). Case-based reasoning: Business applications. *Communications of the ACM 47*(3), 40–42.

Althoff, K.-D. (1992). *Eine fallbasierte Lernkomponente als integrierter Bestandteil der* MOLTKE-*Werkbank zur Diagnose technischer Systeme*. Ph. D. thesis, Dept. of Computer Science, University of Kaiserslautern, Germany.

Althoff, K.-D., E. Auriol, R. Bergmann, S. Breen, S. Dittrich, R. Johnston, M. Manago, R. Traphöner, and S. Wess (1995). Case-based reasoning for decision support and diagnostic problem solving: The INRECA approach. See Bartsch-Spörl, Janetzko, and Wess (1995), pp. 63–72. Available as Report LSA-95-02.

Anderson, J. R. (1983). *The architecture of cognition*. Harvard University Press.

Anderson, J. R. (1988). A spreading activation theory of memory. In A. M. Collins and E. E. Smith (Eds.), *Readings in Cognitive Science*, Chapter 2.4, pp. 137–145. Morgan Kaufmann.

Anderson, J. R. (1989). A theory of the origins of human knowledge. *Artificial Intelligence 40*(1–3), 313–351. Special Volume on Machine Learning.

Ashley, K. D. (1987). *Modelling Legal Argument: Reasoning with Cases and Hypotheticals*. Ph. D. thesis, Department of Computer and Information Science, University of Massachusetts at Amherst.

Ashley, K. D. (1990). *Modeling Legal Argument: Reasoning with Case and Hypotheticals.* Cambridge: MIT-Press.

Auriol, E., K.-D. Althoff, S. Wess, and S. Dittrich (1994). Integrating induction and case-based reasoning: Methodological approach and first evaluations. See Keane, Haton, and Manago (1994), pp. 18–32.

Auriol, E., S. Wess, M. Manago, K.-D. Althoff, and R. Traphöner (1995). INRECA: A seamlessly integrated system based on inductive inference and case-based reasoning. See Aamodt and Veloso (1995), pp. 371–380.

Ballim, A. and N. Karacapilidis (1998). Modeling Discourse Acts in Computer-Assisted Collaborative Decision Making. See Reimer (1998).

Bareiss, R., K. L. Branting, and B. W. Porter (1988). The Role of Explanation in Exemplar-Based Classification and Learning. See Kolodner (1988a).

Barletta, R. and P. Klahr (1997). Effective Knowledge Navigation For Problem Solving Using Heterogeneous Content Types. In *AAAI-97 Artificial Intelligence and Knowledge Management Workshop*. AAAI: AAAI Press.

Barletta, R. and W. Mark (1988). Explanation-Based Indexing of Cases. See AAAI (1988).

Bartlett, R. (1932). *Remembering: A study in experimental and social psychology.* London: Cambridge University Press.

Bartsch-Spörl, B., D. Janetzko, and S. Wess (Eds.) (1995). *3rd German Workshop on CBR: Fallbasiertes Schließen - Grundlagen und Anwendungen* —, Kaiserslautern. University of Kaiserslautern. Available as Report LSA-95-02.

Beam, C. and A. Segev (1996, August). Electronic catalogs and negotiations. CITM Working Paper 96-WP-1016, Fisher Center for Information Technology and Managment, University of California, Berkely. Available at http://haas.berkly.edu/ citm/nego-proj.html.

Beam, C. and A. Segev (1997). Automated negotiations: A survey of the state of the art. CITM Working Paper, Fisher Center for Information Technology and Managment, University of California, Berkely.

Benjamins, V. R., D. Fensel, and A. G. Perez (1998). Knowledge Management through Ontologies. See Reimer (1998).

Bergmann, R. (1996). *Effizientes Problemlösen durch flexible Wiederverwendung von Fällen auf verschiedenen Abstraktionsebenen.* Ph. D. thesis, Universität Kaiserslautern. Available as DISKI 138, infix Verlag.

Bergmann, R., S. Breen, M. Göker, M. Manago, J. Schumacher, A. Stahl, E. Tartarin, S. Wess, and W. Wilke (1998). The INRECA-II Methodology for Building and Maintaining CBR Applications. See Gierl and Lenz (1998), pp. 3–12.

Bergmann, R., H. Muñoz-Avila, M. Veloso, and E. Melis (1998). CBR Applied to Planning. See Lenz, Burkhard, Bartsch-Spörl, and Wess (1998), Chapter 7, pp. 169–200.

Bergmann, R., G. Pews, and W. Wilke (1993). Explanation-based similarity: A unifiying approach for integrating domain knowledge into case-based reasoning for diagnosis and planning tasks. See Wess, Althoff, and Richter (1993), pp. 182–196.

Bergmann, R. and W. Wilke (1995). Building and refining abstract planning cases by change of representation language. *Journal of Artificial Intelligence Research 3*, 53–118.

Bergmann, R. and W. Wilke (Eds.) (1997). *5th German Workshop on CBR — Foundations, Systems, and Applications* —, Report LSA-97-01E, Kaiserslautern. University of Kaiserslautern.

Bergmann, R., W. Wilke, K.-D. Althoff, R. Johnston, and S. Breen (1997). Ingredients for developing a case-based reasoning methodology. See Bergmann and Wilke (1997), pp. 49–58.

Börner, K. (1998). CBR for Design. See Lenz, Burkhard, Bartsch-Spörl, and Wess (1998), Chapter 8, pp. 201–234.

Branting, K. L. (1991). *Integrating Rules and Precedents for Classification and Explanation: Automating Legal Analysis*. Ph. D. thesis, University of Texas at Austin.

Brown, M. G. (1993). An under-lying memory model to support case retrieval. See Wess, Althoff, and Richter (1993), pp. 132–143.

Brown, M. G. (1994). *A memory model for case retrieval by activation passing*. Ph. D. thesis, University of Manchester, Manchester, England.

Brüninghaus, S. and K. D. Ashley (1997). Using Machine Learning for Assigning Indices to Textual Cases. See Leake and Plaza (1997), pp. 303–312.

Burke, R., K. Hammond, and J. Kozlovsky (1996). Knowledge-based Information Retrieval for Semi-Structured Text. In *Working Notes AAAI Fall Symposium AI Applications in Knowledge Navigation and Retrieval*, MIT.

Burke, R., K. Hammond, V. Kulyukin, S. Lytinen, N. Tomuro, and S. Schoenberg (1997a). Natural Language Proessing in the FAQ Finder System: Results and Prospects. In *Working Notes AAAI Spring Symposium NLP for the WWW*, Stanford University, CA.

Burke, R., K. Hammond, V. Kulyukin, S. Lytinen, N. Tomuro, and S. Schoenberg (1997b). Question Answering from Frequently Asked Question Files. *AI Magazine 18*(2), 57–66.

Burke, R. D. (1998). Surveying the Opportunities for Textual CBR: A Position Paper. See Lenz and Ashley (1998), pp. 13–19. AAAI Technical Report WS-98-12.

Burkhard, H.-D. (1995a). Case Retrieval Nets and cognitive modelling. Techn. report, Humboldt University, Berlin. Extended abstract in Workshop "From AI to Cognitive Science, and Back". In L. Dreschler-Fischer, S. Pribbenow (Eds.): KI-95 Activities: Workshops, Posters, Demos. Gesellschaft für Informatik, 1995, pp.53-54.

Burkhard, H.-D. (1995b). Case Retrieval Nets and cognitive modelling (extended abstract). In L. Dreschler-Fischer and S. Pribbenow (Eds.), *KI-95 Activities: Workshops, Posters, Demos*, Bonn, pp. 53–54. Gesellschaft für Informatik.

Burkhard, H.-D. (1998a). Extending some Foundations of CBR. See Lenz, Burkhard, Bartsch-Spörl, and Wess (1998), Chapter 2, pp. 17–50.

Burkhard, H.-D. (1998b). Information Gathering for Vague Queries Using Case Retrieval Nets. In I. Balderjahn, R. Mathar, and S. Martin (Eds.), *Classification, Data Analysis, and Data Highways*, Studies in Classification, Data Analysis, and Knowledge Organization, Berlin, pp. 345–354. Springer Verlag.

Burkhard, H.-D. and M. Lenz (Eds.) (1996). *4th German Workshop on CBR — System Development and Evaluation —*, Informatik-Berichte, Berlin. Humboldt University.

Burstein, M. H. (1989). Analogy vs. CBR: The Purpose of Mapping. See Hammond (1989), pp. 133 – 136.

Busch, K.-H. (1998a, June). Customer Support for Siemens Products. In *Daimler-Benz Technology Workshop Case-Based Reasoning*, Ulm, Germany.

Busch, K.-H. (1998b). Customer Support for Siemens Products on the Internet and CD-ROM. Talk at EWCBR-98 Industry Day.

Carbonell, J. G. (1986). Derivational analogy: a theory of reconstructive problem solving and expertise acquisition. In R. Michalski, J. Carbonnel, and T. Mitchell (Eds.), *Machine*

Learning: an Artificial Intelligence Approach, Volume 2, pp. 371–392. Los Altos, CA: Morgan Kaufmann.

Cardie, C. (1993). A case-based approach to knowledge acquisition for domain-specific sentence analysis. In *Proc. AAAI-93*.

Charniak, E. (1983). Passing markers: A theory of the contextual influence in language comprehension. *Cognitive Science 7*, 171–190.

Cognitive Science Society (1986). *Proceedings of the 8th Annual Conference of the Cognitive Science Society*, Amherst, Massachusetts, USA. Cognitive Science Society.

Cowie, J. and W. Lehnert (1996). Information extraction. *Communications of the ACM 39*(1), 80–91.

Crossley, M., J. Davies, A. McGrath, and M. Reiman-Greene (1998). The Knowledge Garden. See Reimer (1998).

CSS (Ed.) (1991). *Proceedings of the 13th Annual Conference of the Cognitive Science Society*, Chicago, Illinois, USA. Cognitive Science Society.

Cunningham, H. (1997). Information extraction - a user guide. Research Memo CS-97-02, University of Sheffield, Sheffield.

Daniels, J. J. (1997). *Retrieval of Passages for Information Reduction*. Ph. D. thesis, University of Massachusetts at Amherst.

Daniels, J. J. (1998). From Cases to Documents to Passages. See Lenz and Ashley (1998), pp. 8–12. AAAI Technical Report WS-98-12.

Daniels, J. J. and E. L. Rissland (1997). What You Saw Is What You Want: Using Cases to Seed Information. See Leake and Plaza (1997), pp. 325–336.

Domeshek, E. A. (1989). Parallelism for index generation and reminding. See Hammond (1989), pp. 244 – 247.

Elmasri, R. and S. B. Navathe (1994). *Fundamentals of Database Systems*. Redwood City: Benjamin/Cunnings Publishing Company, Inc.

Enzinger, A., F. Puppe, and G. Strube (1994). Problemlösen ohne Suchen? *KI 8*(1), 73–81.

Feldman, R., M. Fresko, H. Hirsh, Y. Aumann, O. Liphstat, Y. Schler, and M. Raiman (1998). Knowledge Management: A Text Mining Approach. See Reimer (1998).

Fensel, D., M. Erdmann, and R. Studer (1998). Ontobroker: The Very High Idea. In *Proceedings of the 11th International Flairs Conference (FLAIRS-98)*, Sanibal Island, FL.

Friedman, J. H., J. L. Bentley, and R. A. Finkel (1977). An algorithm for finding best matches in logarithmic expected time. *ACM Transactions Math. Software 3*, 209–226.

Fuhr, N. (1990). A probabilistic framework for vague queries and imprecise information in databases. In D. McLeod, R. Sacks-Davis, and H. Schek (Eds.), *Proceedings of the 16th International Conference on Very Large Databases*, Los Altos, Cal., pp. 696–707. Morgan Kaufman.

Gebhardt, F., A. Voß, W. Gräther, and B. Schmidt-Belz (1997). *Reasoning with Complex Cases*, Volume 393 of *International Series in Engineering and Computer Science*. Boston: Kluwer Academic Publishers.

Gentner, D. (1983). Structure-mapping: A theorectical framework for Analogy. *Cognitive Science 7*, 155–170.

Gierl, L., M. Bull, and R. Schmidt (1998). CBR in Medicine. See Lenz, Burkhard, Bartsch-Spörl, and Wess (1998), Chapter 11, pp. 273–298.

Gierl, L. and M. Lenz (Eds.) (1998). *Proceedings 6th German Workshop on Case-Based Reasoning*, IMIB Series Vol. 7, Rostock. Inst. fuer Medizinische Informatik und Biometrie, University of Rostock.

Gierl, L. and S. Stengel-Rutkowski (1994). Integrating consultation and semi-automatic knowledge acquisition in a prototype-based architecture: Experiences with dysmorphic syndromes. *Artifical Intelligence in Medicine 6*, 29–49.

Glintschert, A. (1998). ThemeSearch: Aufbau einer intelligenten, themenspezifischen Suchmaschine im WWW. Master's thesis, Humboldt University Berlin.

Göker, M., T. Roth-Berghofer, R. Bergmann, T. Pantleon, R. Traphöner, S. Wess, and W. Wilke (1998). The Development of HOMER: A Case-Based CAD/CAM Help-Desk Support Tool. See Smyth and Cunningham (1998), pp. 346–357.

Gonzalez, A. and R. Laureano-Ortiz (1992). A case-based reasoning approach to real estate property appraisal. *Expert Systems With Applications 4*(2), 229–246.

Gonzalez, A. J. and D. D. Dankel (1993). *The Engineering of Knowledge-Based Systems*. Englewood Cliffs, NJ: Prentice Hall.

Goos, K. (1994). Preselection strategies for case based classification. In B. Nebel and L. Dreschler-Fischer (Eds.), *KI-94: Advances in Artificial Intelligence*, Number 861 in Lecture Notes in Artificial Intelligence, Berlin, pp. 28–38. Springer.

Görz, G. and S. Hölldobler (Eds.) (1996). *KI-96: Advances in Artificial Intelligence*, Lecture Notes in Artificial Intelligence, 1137. Springer Verlag.

Hammond, K. J. (Ed.) (1989). *Proceedings: Case-Based Reasoning Workshop*. Morgan Kaufmann Publishers.

Hayes-Roth, F., D. A. Waterman, and D. B. Lenat (1983). *Building Expert Systems*. New York: Addison-Wesley.

Heckerman, D. (1991). *Probabilistic similarity networks*. Cambridge: MIT Press.

Heider, R. (1996). Troubleshooting CFM 56-3 Engines for Boeing 737 Using CBR and Data Mining. See Smith and Faltings (1996), pp. 512–518.

Herzog, O. and A. Günther (Eds.) (1998). *KI-98: Advances in Artificial Intelligence*, Lecture Notes in Artificial Intelligence, 1504. Springer Verlag.

Hübner, A. (1999). Semiautomatische Wissensanreicherung im Textuellen Fallbasierten Schließen. Master's thesis, Humboldt-Universität zu Berlin.

IJCAI (1983). *Proceedings of the 8th International Conference on Artificial Intelligence IJCAI-83*, Karlsruhe, Germany. IJCAI.

IJCAI (1987). *Proceedings of the 10th International Conference on Artificial Intelligence IJCAI-87*, Milan, Italy. IJCAI.

Kaindl, H. (1989). *Problemlösen durch heuristische Suche*. Wien, New York: Springer Verlag.

Kass, A. K. (1986). Modifying Explanations to Understand Stories. See Cognitive Science Society (1986).

Kass, A. K. (1991). Adaptation Strategies for Case-Based Plan Recognition. See CSS (1991), pp. 161–166.

Keane, M. T., J. P. Haton, and M. Manago (Eds.) (1994). *Second European Workshop on Case-Based Reasoning (EWCBR-94)*. AcknoSoft Press.

King, G. (1998). Using CBR for Sales Support of Electronic Devices. Talk at EWCBR-98 Industry Day.

Kitano, H., A. Shibata, and H. Shimazu (1993). Case-method: A methodology for building large-scale case-based systems. In *Proceedings AAAI-93*.

Kitano, H., A. Shibata, H. Shimazu, J. Kajihara, and A. Sato (1992). Building Large-Scale Corporate-Wide Case-Based Systems: Integration of organizational and machine Executable Algorithms. In *Proceedings of the AAAI-92*.

Kolodner, J. L. (1983a). Maintaining Organization in a Dynamic Long-Term Memory. *Cognitive Science 7*, 243–280.

Kolodner, J. L. (1983b). Reconstructive Memory : A Computer Model. *Cognitive Science 7*, 281–328.

Kolodner, J. L. (Ed.) (1988a). *Proceedings Case-Based Reasoning Workshop*. Morgan Kaufmann Publishers.

Kolodner, J. L. (1988b). Retrieving Events from a Case Memory: A Parallel Implementation. See Kolodner (1988a), pp. 233–249.

Kolodner, J. L. (1993). *Case-Based Reasoning*. San Mateo: Morgan Kaufmann.

Krulwich, B. and C. Burkey (1997). The InfoFinder Agent: Learning User Interests through Heuristic Phrase Extraction. *IEEE Expert 12*(5), 22–27.

Kunze, M. and A. Hübner (1998). CBR on Semi-structured Documents: The ExperienceBook and the FAllQ Project. In *Proceedings 6th German Workshop on CBR*.

Ladewig, H. (1994). Expertensysteme für die Wertermittlung von Immobilien. *Grundstücksmarkt und Grundstückswert 4*, 211–217.

Ladewig, H. (1995). Expertisen aus dem Automaten. *Immobilien-Manager 3*, 6–12.

Lange, T. E. (1995). A structured connectionist approach to inferencing and retrieval. In R. Sun and L. A. Bookman (Eds.), *Computational Architechtures Integrating Neural and Symbolic Processes. A perspective on the State of the Art*, Chapter 3, pp. 69–115. Kluwer Academic Publishers.

Lange, T. E. and C. M. Wharton (1993). Dynamic memories: Analysis of an integrated comprehension and episodic memory retrieval model. In *Proceedings 13th International Joint Conference On Artificial Intelligence*, San Mateo, pp. 208–213. Morgan Kaufmann.

Leake, D. B. (1990). *Evaluating explanations*. Ph. D. thesis, Yale University, Computer Science Department. Technical Report 769.

Leake, D. B., A. Kinley, and D. Wilson (1995). Learning to improve case adaptation by introspective reasoning and cbr. See Aamodt and Veloso (1995), pp. 229–240.

Leake, D. B. and E. Plaza (Eds.) (1997). *Case-Based Reasoning Research and Development, Proceedings ICCBR-97*, Lecture Notes in Artificial Intelligence, 1266. Springer Verlag.

Lee, S. K. (1992). An Extended Relational Database Model for Uncertain And Imprecise Information. In *Proceedings VLDB92*, Vancouver, Canada, pp. 211–220.

Lenz, M. (1993). CABATA - A hybrid CBR system. In *Pre-prints First European Workshop on Case-Based Reasoning*, pp. 204–209. University of Kaiserslautern.

Lenz, M. (1994a). Case-based reasoning for holiday planning. In W. Schertler, B. Schmid, A. M. Tjoa, and H. Werthner (Eds.), *Information and Communications Technologies in Tourism*, pp. 126–132. Springer Verlag.

Lenz, M. (1994b). Fallbasiertes Schließen in schwach strukturierten Domänen. Master's thesis, Humboldt-Universität zu Berlin.

Lenz, M. (1996a). Case Retrieval Nets applied to large case bases. See Burkhard and Lenz (1996), pp. 111–118.

Lenz, M. (1996b). IMTAS: Intelligent Multimedia Travel Agent System. In S. Klein, B. Schmid, A. M. Tjoa, and H. Werthner (Eds.), *Information and Communications Technologies in Tourism*, pp. 11–17. Wien, New York: Springer Verlag.

Lenz, M. (1998a). Defining Knowledge Layers for Textual Case-Based Reasoning. See Smyth and Cunningham (1998), pp. 298–309.

Lenz, M. (1998b). Experiences from Deploying Electronic Commerce Applications. Techn. report, Humboldt University, Berlin.

Lenz, M. (1998c). Knowledge Sources for Textual CBR Applications. See Lenz and Ashley (1998), pp. 24–29. AAAI Technical Report WS-98-12.

Lenz, M. (1998d). Textual CBR and Information Retrieval – A Comparison. See Gierl and Lenz (1998).

Lenz, M. (1999). Experiences from deploying cbr applications in electronic commerce. In *accepted for GWCBR-99*.

Lenz, M. and K. D. Ashley (Eds.) (1998). *Proceedings of the AAAI-98 Workshop on Textual Case-Based Reasoning*. AAAI. AAAI Technical Report WS-98-12.

Lenz, M., E. Auriol, and M. Manago (1998). Diagnosis and Decision Support. See Lenz, Burkhard, Bartsch-Spörl, and Wess (1998), Chapter 3, pp. 51–90.

Lenz, M. and H.-D. Burkhard (1995). Retrieval ohne Suche? See Bartsch-Spörl, Janetzko, and Wess (1995), pp. 1–10. Available as Report LSA-95-02.

Lenz, M. and H.-D. Burkhard (1996a). Case Retrieval Nets: Foundations, properties, implementation, and results. Techn. report, Humboldt University, Berlin.

Lenz, M. and H.-D. Burkhard (1996b). Lazy propagation in Case Retrieval Nets. See Wahlster (1996), pp. 127–131.

Lenz, M. and H.-D. Burkhard (1997a). Applying CBR for Document Retrieval. In Y. Nakatani (Ed.), *Proceedings Workshop Practical Use of CBR at IJCAI-97*, pp. 75–86.

Lenz, M. and H.-D. Burkhard (1997b). CBR for Document Retrieval - The FALLQ Project. See Leake and Plaza (1997), pp. 84–93.

Lenz, M., H.-D. Burkhard, B. Bartsch-Spörl, and S. Wess (1998). *Case-Based Reasoning Technology – From Foundations to Applications*. Lecture Notes in Artificial Intelligence 1400. Springer Verlag.

Lenz, M., H.-D. Burkhard, and S. Brückner (1996). Applying Case Retrieval Nets to diagnostic tasks in technical domains. See Smith and Faltings (1996), pp. 219–233.

Lenz, M., H.-D. Burkhard, P. Pirk, E. Auriol, and M. Manago (1996). CBR for Diagnosis and Decision Support. *AI Communications 9*(3), 138–146.

Lenz, M. and A. Glitschert (1999). On texts, cases, and concepts. In F. Puppe (Ed.), *to appear in: Proc. XPS-99*, Lecture Notes in Artificial Intelligence, pp. ???–??? Springer Verlag.

Lenz, M. and H. Ladewig (1996). Fallbasierte Unterstützung bei der Immobilienbewertung. *Wirtschaftsinformatik, Schwerpunktheft Fallbasierte Entscheidungsunterstützung 38*(1), 39–42.

Lewis, D. D. and K. Sparck Jones (1996). Natural language processing for information retrieval. *Communications of the ACM 39*(1), 92–101.

Mahe, S. and C. Rieu (1998). A Pull Approach to Knowledge Management. See Reimer (1998).

Meadow, C. T. (1991). *Text Information Retrieval Systems*. San Diego, CA: Academic Press.

Miller, G. A. (1995). WordNet: A lexical database for english. *Communications of the ACM 38*(11), 39–41.

Motro, A. (1988). VAGUE: A User Interface to Relational Databases that Permits Vague Queries. *ACM Transactions on Office Information Systems 6*(3), 187–214.

Murphy, P. M. and D. W. Aha (1992). *UCI Repository of Machine Learning Databases and Domain Theories*. Irvine, CA: University of California.

Nakata, K., A. Voss, M. Juhnke, and T. Kreifelts (1998). Collaborative Conecpt Extraction from Documents. See Reimer (1998).

Newell, A. and H. A. Simon (1972). *Human Problem Solving*. Englewood Cliffs, NJ: Prentice-Hall.

Nilsson, N. J. (1982). *Principles of Artificial Intelligence*. Berlin, Heidelberg, New York: Springer Verlag.

O'Leary, D. E. (1998a). Knowledge Management Systems: Converting and Connecting. *IEEE Intelligent Systems 13*(3), 30–33.

O'Leary, D. E. (1998b). Using AI in Knowledge Management: Knowledge Bases and Ontologies. *IEEE Intelligent Systems 13*(3), 34–39.

Pearl, J. (1986). Fusion, propagation, and structuring in belief networks. *Artificial Intelligence 29*, 241–288.

Pearl, J. (1988). *Probabilistic reasoning in intelligent systems: Networks of plausible inference*. San Mateo: Morgan Kaufmann.

Porter, M. F. (1980). An algorithm for suffix stripping. *Program 14*, 130–137.

Puppe, F. (1993). *Systematic Introduction to Expert Systems*. Springer Verlag.

Purvis, L. and P. Pu (1995). Adaptation using constraint satisfaction techniques. See Aamodt and Veloso (1995), pp. 289–300.

Quinlan, J. R. (1990). Induction of decision trees. In J. W. Shavlik and T. G. Dietterich (Eds.), *Readings in Machine Learning*, Chapter 2.2.1, pp. 57–69. San Mateo: Morgan Kaufmann.

Quinlan, R. J. (1993). *C4.5 Programs for Machine Learning*. Morgan Kaufmann.

Reimer, U. (Ed.) (1998). *Practical Aspects of Knowledge Management (PAKM-98)*.

Richter, M. M. (1992). *Prinzipien der Künstlichen Intelligenz*. Stuttgart, Germany: B. G. Teubner.

Richter, M. M. (1995). The knowledge contained in similarity measures. Invited Talk at ICCBR-95. http://wwwagr.informatik.uni-kl.de/~lsa/CBR/Richtericcbr95remarks.html.

Richter, M. M. (1998). Introduction. See Lenz, Burkhard, Bartsch-Spörl, and Wess (1998), Chapter 1, pp. 1–16.

Riesbeck, C. K. and R. C. Schank (1989). *Inside Case-Based Reasoning*. Hillsdale, NJ: Lawrence Erlbaum Associates.

Rijsbergen, C. J. v. (1979). *Information Retrieval* (2 ed.). London: Butterworth-Heinemann.

Riloff, E. (1991). Automatically constructing a dictionary for information extraction tasks. See AAAI (1991).

Riloff, E. and W. Lehnert (1994). Information extraction as a basis for high-precision text classification. *ACM Transactions on Information Systems 12*(3), 296–333.

Rissland, E. L. (1983). Examples in Legal Reasoning: Legal Hypotheticals. See IJCAI (1983).

Rissland, E. L. and K. D. Ashley (1987). HYPO: A Case-Based Reasoning System. See IJCAI (1987).

Rissland, E. L., D. B. Skalak, and M. T. Friedman (1993). Case retrieval through multiple indexing and heuristic search. In *Proceedings 13th International Joint Conference On Artificial Intelligence*, pp. 902–908.

Röpnack, A., M. Schindler, and T. Schwan (1998). Concepts of the Enterprise Knowledge Medium. See Reimer (1998).

Rumelhart, D. E., J. L. McClelland, and the PDP Research Group (1986). *Parallel Distributed Processing: Explorations in the Microstructure of Cognition. Volume 1: Foundations.* Cambridge: MIT Press.

Russel, S. J. and P. Norvig (1995). *Artificial Intelligence - A Modern Approach.* Englewood Cliffs, NJ: Prentice Hall.

Salton, G. and M. McGill (1983). *Introduction to Modern Information Retrieval.* New York: McGraw-Hill.

Schaaf, J. (1998). *Über die Suche nach situationsgerechten Fällen im fallbasierten Schließen.* DISKI 179, Universität Kaiserslautern.

Schaaf, J. W. (1994). Using gestalten to retrieve cases. In M. Keane, J.-P. Haton, and M. Manago (Eds.), *EWCBR-94: Second European Workshop on Case-Based Reasoning*, Chantilly, pp. 75–83. AcknoSoft Press, Paris.

Schaaf, J. W. (1995). Fish and Sink: an anytime-algorithm to retrieve adequate cases. See Aamodt and Veloso (1995), pp. 538–547.

Schaaf, J. W. (1996). Fish and shrink: a next step towards efficient case retrieval in large scaled case bases. See Smith and Faltings (1996), pp. 362–376.

Schank, R. C. (1982). *Dynamic Memory: A Theory of Learning in Computers and People.* New York: Cambridge University Press.

Schmidt, R., L. Boscher, B. Heindl, G. Schmid, B. Pollwein, and L. Gierl (1995). Adaptation and abstraction in a case-based antibiotics therapy adviser. In P. Barahona, M. Stefanelli, and J. Wyatt (Eds.), *Proceedings of the 5th Conference on Artificial Intelligence in Medicine Europe (AIME-95)*, Lecture Notes in Artificial Intelligence, 934, pp. 209–217. Springer Verlag.

Schumacher, J., W. Wilke, and B. Smyth (1998). Domain independent adaptation using configuration techniques. See Gierl and Lenz (1998).

Shortliffe, E. H. (1976). *Computer-Based Medical Consultation: MYCIN.* New York: Elsevier.

Smith, I. and B. Faltings (Eds.) (1996). *Advances in Case-Based Reasoning*, Lecture Notes in Artificial Intelligence, 1186. Springer Verlag.

Smyth, B. and P. Cunningham (Eds.) (1998). *Advances in Case-Based Reasoning*, Lecture Notes in Artificial Intelligence, 1488. Springer Verlag.

Smyth, B. and M. T. Keane (1993). Retrieving adaptable cases: The role of adaptation knowledge in case retrieval. See Wess, Althoff, and Richter (1993), pp. 209–220.

Smyth, B. and M. T. Keane (1996). Using adaptation knowledge to retrieve and adapt design cases. *Journal of Knowledge Based Systems 9*, 127–135.

Soderland, S. G. (1997). *Learning Text Analysis Rules for Domain-Specific Natural Language Processing.* Ph. D. thesis, University of Massachusetts Amherst, USA.

Soderland, S. G., D. Fisher, J. Aseltine, and W. Lehnert (1995). CRYSTAL: Inducing a Conceptual Dictionary. In *Proc. IJCAI-95.*

Stanoevska-Slabeva, K., A. Hombrecher, S. Handschuh, and B. Schmid (1998). Efficient Information Retrieval: Tools for Knowledge Management. See Reimer (1998).

Sun, R. (1995a). Robust reasoning: integrating rule-based and similarity-based reasoning. *Artificial Intelligence 75*(2), 241–295.

Sun, R. (1995b). A two-level hybrid architecture for structuring knowledge for commonsense reasoning. In R. Sun and L. A. Bookman (Eds.), *Computational Architechtures Integrating Neural and Symbolic Processes. A perspective on the State of the Art*, Chapter 8, pp. 247–280. Kluwer Academic Publishers.

Sutcliffe, R. F. (1992). Representing meaning using microfeatures. In R. G. Reilly and N. E. Sharkey (Eds.), *Connectionist Approaches to Natural Language Processing*. Lawrence Earlbaum Associates.

Thagard, P. and K. J. Holyoak (1989). Why Indexing is the Wrong Way to Think About Analog Retrieval. See Hammond (1989), pp. 36–40.

Thagard, P., K. J. Holyoak, G. Nelson, and D. Gochfeld (1990). Analog retrieval by constraint satisfaction. *Artificial Intelligence 46*, 259–310.

Tsatsoulis, C., Q. Cheng, and H.-Y. Wei (1997). Integrating Case-Based Reasoning and Decision Theory. *IEEE Expert 12*(4), 46–55.

Ullman, J. D. (1988). *Principles of Database and Knowledgebase Systems*. Rockville, Maryland: Computer Science Press.

Veloso, M., H. Muñoz-Avila, and R. Bergmann (1996). Case-based planning: selected methods and systems. *AI Communications 9*(3), 128–137.

Veloso, M. M. (1994). *Planning and Learning by Analogical Reasoning*. Number 886 in Lecture Notes in Computer Science. Berlin: Springer.

Voß, A. and C.-H. Coulon (1996). Structural adaptation with TOPO. In A. Voß (Ed.), *ECAI 96, W[orkshop] 6, Adaptation inCase-Based Reasoning*, pp. 52–54. ECAI 96.

Wahlster, W. (Ed.) (1996). *Proceedings 12th European Conference on Artificial Intelligence ECAI-96*. John Wiley and Sons.

Walsh, J. and G. Ungson (1991). Organizational Memory. *Academy of Management Review 16*, 57–91.

Waltz, D. (1989a). Is indexing used for retrieval? See Hammond (1989), pp. 41 – 44.

Waltz, D. (1989b). Panel dicussion on "indexing algorithhms". See Hammond (1989), pp. 25 – 30.

Wess, S. (1995). *Fallbasiertes Problemlösen in wissensbasierten Systemen zur Entscheidungsunterstützung und Diagnostik*. Ph. D. thesis, Universität Kaiserslautern. Available as DISKI 126, infix Verlag.

Wess, S., K.-D. Althoff, and G. Derwand (1993). Using kd-trees to improve the retrieval step in case-based reasoning. See Wess, Althoff, and Richter (1993), pp. 167–181.

Wess, S., K.-D. Althoff, and M. M. Richter (Eds.) (1993). *Topics in Case-Based Reasoning. Proceedings of the First European Workshop on Case-Based Reasoning (EWCBR-93)*, Lecture Notes in Artificial Intelligence, 837. Springer Verlag.

Wilke, W. (1997). Case-based reasoning and electronic commerce. Invited Talk at ICCBR-97.

Wilke, W., R. Bergmann, and S. Wess (1998). Negotiation During Intelligent Sales Support with Case-Based Reasoning. See Gierl and Lenz (1998), pp. 213–222.

Wilke, W., M. Lenz, and S. Wess (1998). Intelligent Sales Support with CBR. See Lenz, Burkhard, Bartsch-Spörl, and Wess (1998), Chapter 4, pp. 91–114.

Wilke, W., B. Smyth, and P. Cunningham (1998). Using Configurations Techniques for Adaptation. See Lenz, Burkhard, Bartsch-Spörl, and Wess (1998), Chapter 6, pp. 139–168.

Wolverton, M. (1995). An investigation of marker-passing algorithms for analogue retrieval. See Aamodt and Veloso (1995), pp. 359–370.

Wolverton, M. and B. Hayes-Roth (1994). Retrieving semantically distant analogies with knowledge-directed spreading activation. See AAAI (1994), pp. 56–61.

Index

Appendix A

Evaluation Data

A.1 Evaluation Queries

On the following pages, the original queries (O) and their rephrased (R) versions used for the evaluation in Section 9.4 are listed. Due to the underlying document collection, this evaluation has been on German documents. In Section A.4, this experimental data will be translated into English.

Table A.1: List of queries used for evaluation of the SKM

Query No.1
- O: WIE KANN MIT WINCC EINE MESSAGE-BOX AUFGERUFEN WERDEN, DIE IM VORDERGRUND UND MODAL IST?
- R: PROGRAMM FUER EINE MODALE EREIGNIS-BOX SO DASS SIE IM VORDERGRUND VON WIN CC ERSCHEINT.

Query No.2
- O: WAS PASSIERT, WENN ICH EINEN TIMER MIT DER ZEITDAUER 0 S STARTE?
- R: WIE IST DER ZUSTAND DER 945'ER CPU WENN MAN DEN TIMER MIT DAUER VON NULL MILLISEKUNDEN STARTET?

Query No.3
- O: WAS FUER VORTEILE HAT EINE WORTWEISE FLANKENAUSWERTUNG UND WIE MUSS ICH SIE PROGRAMMIEREN?
- R: WOZU DIENT DIE WORTWEISE AUSWERTUNG VON FLANKEN IN PROGRAMMEN?

Query No.4
- O: KANN ICH MIT DEM COM115F FEHLEREINTRAEGE AUSLESEN, OHNE DABEI DIE CPU STOPPEN ZU MUESSEN?
- R: IN WELCHEM ZUSTAND KOENNEN BEI S5-115F FEHLEREINTRAEGE GELESEN WERDEN? GEHT DAS IM RUN-ZUSTAND?

Query No.5
- O: WIE MUSS ICH DIE OBER- UND UNTERGRENZE BEIM PARAMETRIEREN DES FB 31 (RLG:AE) VORGEBEN?
- R: WELCHE OBERGRENZEN UND UNTERGRENZEN FUER TEMPERATURBEREICHE GELTEN FUER DIE PARAMETRIERUNG DES FB 31?

Query No.6
- O: DIE BAUGRUPPEN 422-8MA, 482-8MA UND 451-8MR, SOWIE DIE ONBOARD-EIN-/AUSGAENGE DES AG 95 WERDEN UEBER FRONTSTECKER ANGESCHLOSSEN. WELCHEN ADERQUERSCHNITT DARF ICH VERWENDEN, WENN ICH DEN FRONTSTECKER MIT SCHRAUBANSCHLUSS VERWENDE?
- R: WIEVIEL KABELQUERSCHNITT MUESSEN DIE ADERN BEI FRONTSTECKERN FUER S5-100U BEI BG-TYP 428 HABEN?

Query No.7
- O: ICH HABE PRO TOOL 2.51 UND STEP 7 VERSION 3.1 INTEGRIERT INSTALLIERT. WORAN KANN ES LIEGEN, DASS ICH KEINE OP-PROJEKTIERUNGEN MEHR AUS DEM PROJEKT-MANAGER LOESCHEN KANN?
- R: WELCHE INSTALLATIONSHINWEISE GIBT ES FUER PROTOOL? NACH INSTALLATION VON VERSION 2.51 KANN MAN DIE ANGABEN ZUM PROJEKT AUS DEM OPERATOR-PANEL NICHT LOESCHEN.

Query No.8
- O: ICH MOECHTE MEINEN DP-SLAVE AN EINEN DP-MASTER ANSCHLIESSEN. MIR FEHLT JEDOCH DIE TYP- BZW. DIE GSD-DATEI FUER DIESEN DP-SLAVE. WO KANN ICH DIESE TYP/ GSD-DATEI BEZIEHEN?
- R: ICH BENOETIGE INFORMATIONEN ZUM HERUNTERLADEN VON TYPDATEIEN FUER DIE ANKOPPLUNG VON DP-SLAVESCHNITTSTELLE IN EINEM DP-MASTER-SYSTEM.

Query No.9
- O: IST ES MOEGLICH EINE ELEKTRISCHE REDUNDANZ ZWISCHEN OLM S3/4 UND OLM P3/4 AUFZUBAUEN?
- R: WELCHE AUFBAUMOEGLICHKEITEN GIBT ES FUER ELEKTRISCHE REDUNDANZ BEI EINEM OPTICAL LINK MODULE?

Query No.10
- O: IST ES MOEGLICH DIE CP5412 A2 MIT DEM SOFTWARE-PAKET DP-5412/MS-DOS, WINDOWS ALS SLAVE IN EINEM DP-SYSTEM ZU BETREIBEN?
- R: WELCHE SONDERTREIBER BRAUCHT MAN UM UNTER NT4.0 EINE CP5412 IN EINEM NETZ MIT DP UND PROFIBUS ZU BENUTZEN?

Query No.11
- O: BEI DER INSTALLATION DES DP-5412 PAKETES (6GK1702-5DA00-0EA0, V1.10) IST FOLGENDES PROBLEM AUFGETRETEN: UNTER WIN 3.11 WIRD ZUM ABSCHLUSS DER INSTALLATION DIE MELDUNG "FALSCHE SYSTEMVERSION, BITTE INSTALLIEREN SIE WINDOWS NEU" AUSGEGEBEN. FOLGT MAN DIESER ANWEISUNG, AENDERT SICH TROTZDEM NICHTS. WAS SOLL ICH TUN?
- R: ICH HABE DAS DP 5412 PAKET INSTALLIERT UND DANACH ERSCHIEN DIE MELDUNG "FALSCHE SYSTEMVERSION". WAS SOLL DAS BEDEUTEN?

Query No.12
- O: WIE WIRD DIE SIGNALBAUGRUPPE EINGESTELLT? WERDEN DIE BEREITS EXISTIERENDEN MELDUNGEN UEBERSCHRIEBEN? WIE WIRD DIE SIGNALBAUGRUPPE QUITTIERT?
- R: WELCHE QUITTIERARTEN KOENNEN IN SIGNAL-BG EINGESTELLT WERDEN? WELCHE MELDUNGSQUITTUNG KANN MAN BENUTZEN?

Query No.13
- O: DIE ACHSE LAEUFT SOFORT NACH DEM EINSCHALTEN LOS (ACHSE MACHT EINEN RUCK) UND ICH BEKOMME EINE FEHLERMELDUNG: "DIAGNOSEALARM: 0X010C STILLSTANDSBEREICH". WAS IST DIE FEHLERURSACHE?
- R: WAS BEDEUTET DER FEHLER "DIAGNOSEALARM: 0X010C STILLSTANDSBEREICH" BEI DER ACHSE DES FM354?

Query No.14
- O: AUF MEINER ENGINEERING-STATION (ES) KANN ICH IM SIMATIC-MANAGER KEIN OS-PROJEKT ANLEGEN BZW. ES WIRD KEIN OS-PROJEKT ANGEZEIGT, OBWOHL ES VORHANDEN IST.
- R: DER SIMATIC-MANAGER LEGT KEINE PROJEKTE FUER DIE BEDIENSTATION AN.

Query No.15
- O: WIE FUNKTIONIERT DIE BASISKOMMUNIKATION BEI EINER S7 CPU MIT GLOBALDATEN UND AUF WAS MUSS ICH DABEI ACHTEN?
- R: WIE IST DIE FUNKTION FUER DIE GLOBALDATEN-KOMMUNIKATION ZWISCHEN ZENTRAL-BG REALISIERT?

Query No.16
- O: ICH KANN MIT DEM FB192 NUR EINGAENGE LESEN. DAS SCHREIBEN VON AUSGAENGEN FUEHRT ZU KEINEM FEHLER, ES WERDEN ABER AN DEN SLAVES KEINE AUSGAENGE AUSGEBEN, WARUM?
- R: WIE LIEST MAN MIT DEM FB192 EINGAENGE UND ADRESSIERT FEHLERFREI AUSGAENGE?

Query No.17
- O: DIE CPU 928 GEHT SPORADISCH MIT WECKFEHLER IN STOP
- R: BEI WIEVIEL MILLISEKUNDEN KANN MAN OHNE WECKFEHLER MIT CPU928 ARBEITEN?

Query No.18
- O: WIE KANN COM ET200 V4.X AUF EINEM PG720 INSTALLIERT WERDEN?
- R: ICH WILL AUF PG 720 DIE VERSION 4 EINER COM ET 200 INSTALLIEREN. WIE GEHT DAS?

Query No.19
- O: WELCHEN MODUS MUSS MAN BEI 80486 PROZESSOREN IM SETUP FUER DEN CACHE EINSTELLEN?
- R: ICH HABE EINEN 80486/DX4-PROZESSOR. WELCHER MODE FUER DEN CACHE-SPEICHERBEREICH MUSS BEIM SETUP EINGESTELLT WERDEN?

Query No.20
- O: PRODUKTINFORMATION (C79000-Z8563-C122-02) UEBER SCHALTERSTELLUNGEN DES CP 524
- R: ICH BRAUCHE INFORMATIONEN ZU CP524 PRODUKTEN, INSBESONDERE ZUR STELLUNG DER SCHALTER

Query No.21
- O: WIE KANN MAN STEP7 MICRO/DOS V1.X PROJEKTE IN STEP7 MICRO/WIN V2.0 IMPORTIEREN?
- R: WIE ERFOLGT DER IMPORT VON PROJEKTEN AUS STEP7 MICRO/DOS V1.X IN MICRO/WIN V2.0?

Query No.22
O: WARUM WIRD DER WINKEL UNTER WINNT 4.0 AN RUNDECKEN VERZOGEN?
R: DER WINKEL AN DEN RUNDEN ECKEN WIRD UNTER WINDOWS NT4.0 FALSCH DARGESTELLT.
Query No.23
O: WARUM WIRD DAS WINCC LOGO FARBLICH TOTAL VERFAELSCHT AUSGEDRUCKT?
R: DIE FARBEN BEIM LOGO ZU WIN CC ERSCHEINEN BEIM AUSDRUCKEN VERFAELSCHT. MUSS ICH EINE TREIBERINSTALLATION VORNEHMEN?
Query No.24
O: WAS SIND DIE VORAUSSETZUNGEN FUER DEN CLIENT-SERVER BETRIEB BEI WINCC?
R: WAS MUSS MAN BEACHTEN, WENN MAN WINCC IN NETZWERKEN BETREIBEN WILL.
Query No.25
O: -
R: IST DIE LSC COROS-STATION Y2K-GETESTET?

A.2 Precision–Recall Tables for the Different Methods of Normalization

This is the data underlying Figure 9.4 on page 141.

Table A.2: Precision and recall for normalization based on the size of the query (left), based on the case (middle), and on both (right)

Recall	Precision	Recall	Precision	Recall	Precision
0.06	1	0	1	0	1
0.06	0.98	0	1	0	1
0.16	0.98	0	1	0	1
0.32	0.973333	0	1	0	1
0.48	0.95	0	1	0	1
0.54	0.873333	0	1	0	1
0.6	0.767778	0	1	0	1
0.82	0.778667	0	1	0	1
0.88	0.671684	0	1	0	1
0.88	0.510656	0	1	0	1
0.88	0.272322	0	1	0	1
0.92	0.16292	0	0.96	0	1
0.96	0.13302	0	0.96	0	1
0.96	0.0758948	0.04	0.96	0	1
0.96	0.0562105	0.08	0.88	0	1
0.96	0.056	0.16	0.7	0	1
0.96	0.056	0.24	0.504667	0.08	1
0.96	0.056	0.56	0.258752	0.12	1
0.96	0.056	0.62	0.0664971	0.24	0.94
0.96	0.056	0.7	0.038	0.64	0.547125

A.3 Precision–Recall Tables for the Ablation Study

This data corresponds to the results discussed on page 143 and displayed in Figure 9.6.

Table A.3: Precision and recall depending on the amount of knowledge utilized
Top left: Simple keyword search
Top middle: Simple keyword search plus *topic*
Top right: Representation by means of IEs
Bottom left: Representation by means of IEs plus *topic*
Bottom right: Complete similarity measure

Recall	Precision
0	1
0	1
0	1
0.04	0.92
0.12	0.856667
0.12	0.570714
0.22	0.432
0.4	0.274333
0.48	0.084
0.52	0.056

Recall	Precision
0	1
0	1
0.04	0.96
0.04	0.96
0.08	0.97
0.36	0.790667
0.58	0.411159
0.72	0.197556
0.8	0.1
0.8	0.092

Recall	Precision
0	1
0	1
0.22	0.906667
0.34	0.793333
0.58	0.703206
0.76	0.503333
0.82	0.305
0.86	0.143095
0.86	0.096
0.86	0.096

Recall	Precision
0	1
0	1
0	1
0.04	1
0.12	0.94
0.54	0.765873
0.72	0.608762
0.86	0.248
0.86	0.105111
0.86	0.096

Recall	Precision
0.06	1
0.16	0.98
0.48	0.95
0.6	0.767778
0.88	0.674048
0.88	0.290778
0.96	0.164889
0.96	0.112
0.96	0.112
0.96	0.112

A.4 Evaluation Queries

In this section,we present the English translation of the above test set. Note that some of the major problems concerned with German are not reflected in this translation. For example, composite nouns are used frequently in German texts (even with various ways of composing a noun). Also, there is no general rule for generating word forms such that stemming is much more difficult.

Table A.4: English translations of queries used for evaluation of the SKM

Query No.1
- O: How can a message box be called with WinCC which is modal and in the foreground?
- R: Program for a modal event box such that it appears in the foreground of Win CC.

Query No.2
- O: What happens when I start a time with a vitual time of 0 s?
- R: What is the state of a 945 CPU after a timer has been started with duration of zero milliseconds?

Query No.3
- O: What are the advantages of a word-wise edge evaluation and how can this be programmed?
- R: What is word-wise edge detection good for?

Query No.4
- O: Using the Com115F can I read error codes without having to stop the CPU?
- R: In what state can error codes be read in a S5-115F? Is that possible in run mode?

Query No.5
- O: How do I have to set the upper and lower limits when parametrizing the FB 31 (RLG:AE)?
- R: Which limits are valid for the temperature range parameters for a FB 31?

Query No.6
- O: The components 422-8MA, 482-8MA and 451-8MR as well as the onboard I/O of the AG 95 have to be connected via front connectors. Which cable cross-section can I use when applying the front connectors with a screw-type terminal?
- R: What is the cross-section of the strand for front connectors for S5-100U and BG of type 428?

Query No.7
- O: I have installed Pro Tool 2.51 and Step 7 V3.1 (integrated). What may be the reason that I can no longer delete OP project plans from the project manager?
- R: Which hints do you have for installing ProTool? After having installed version 2.51 I cannot delete the specifications for the project from the operator panel.

Query No.8
- O: I want to connect my DP slave with a DP master but I do not have the type / GSD file. Where can I get it?
- R: I need information on how to download type files for connecting DP slave interfaces within a DP master system.

Query No.9
- O: Is it possible to construct an electrical redundancy between OLM S3/4 and OLM P3/4?
- R: Which construction possibilities do exist for an electrical redundancy in optical link modules?

Query No.10
- O: Is it possible to run the CP5412 A2 with the DP-5412/MS-DOS/Windows software as a slave in a DP system?
- R: Which special drivers do I need to use a CP5412 in a net with DP and Profibus unter NT4.0?

Query No.11
- O: When installing the DP-5412 package (6GK1702-5DA00-0EA0, V1.10) I encountered the following problem: Under Win 3.11 the message "Wrong system version, please reinstall Windows." appears after installation. Even when doing so, the behavior remains. What to do?
- R: I have installed the DP 5412 package and after that the message "Wrong system version, please reinstall Windows." appeared. What does that mean?

Query No.12
- O: How is the signal module adjusted? Will the existing messages be overwritten? How is the signal module acknowledged?
- R: Which quitting settings can be chosen in signal modules? Which message acknowledgment can be used?

Query No.13
- O: The axis starts to move heavily immediately after turning on and an error message "Diagnostic alarm: 0X010C stoppage area" appears. What is the reason?
- R: What does the failure "Diagnostic alarm: 0X010C stoppage area" mean for the axis of a FM354?

Query No.14
- O: On my engineering station I cannot build an OS project within the Simatic manager resp. no OS project is displayed even though it is there.
- R: The Simatic manager does not put up projects for the operator station.

Query No.15
- O: How does the basic communication work with a S7 CPU and global data, what do I have to pay attention to?
- R: How is the function for global data communication between CPUs implemented?

Query No.16
- O: With the FB192 I can only read inputs. Writing of outputs does not result in errors but no outputs appear at the slaves. Why?
- R: How do I correctly read inputs and address outputs with the FB192?

Query No.17
- O: The CPU 928 sporadically turns in stop with watch dog failure.
- R: At how many milliseconds can I correctly work with a CPU928 without watch dog failures?

Query No.18
- O: How can a COM ET200 V4.X be installed at a PG720?
- R: I want to install version 4 of a COM ET 200 on PG 720. How can I do that?

Query No.19
O: Which mode do I have to select during setup for the cache in case of 80486 processors?
R: I have a 80486/DX4 processor. Which mode for the cache memory has to be set during setup?
Query No.20
O: Product information (C79000-Z8563-C122-02) about switch positions of the CP 524.
R: I need information about CP524 products, in particular about adjustment of switches.
Query No.21
O: How can I import Step7 Micro/Doc V1.x projects in Step7 Micro/Win V2.0?
R: What to do to import projects from Step7 Micro/Doc V1.X to Micro/Win V2.0?
Query No.22
O: WARUM WIRD DER WINKEL UNTER WINNT 4.0 AN RUNDECKEN VERZOGEN?
O: Why is the angel at round corners distorted unter WinNT4.0?
R: The angel of circular edges is being displayed wrongly under Windows NT4.0.
Query No.23
O: Why is the WinCC logo printed in wrong colors?
R: The colors of the logo of Win CC appear to be wrong. Do I have to install a driver?
Query No.24
O: What are the requirements for client server mode at WinCC?
R: What do I have to consider when using WinCC in networks?
Query No.25
O: -
R: Has the LSC coros station been tested for y2k?