

# 13. Related Areas

Gerd Kamp, Steffen Lange, Christoph Globig

## 13.1 Introduction

As already mentioned in the introduction of this book, case-based reasoning has its origins in cognitive science and artificial intelligence:

- Cognitive science was looking for a model of human problem solving.
- Artificial Intelligence was searching for means to overcome the shortcomings of rule-based expert system for generic and effective problem solving methods.

Because of that interdisciplinary tradition and the generality of the approach, case-based reasoning found widespread use in a great variety of application domains. The previous chapters gave an impressive demonstration of this fact. We have seen the integration of many different techniques and methods with the case-based reasoning approach. It has become clear that CBR is a generic methodology for building knowledge-based systems, rather than an isolated technique that is capable of solving only very specific tasks.

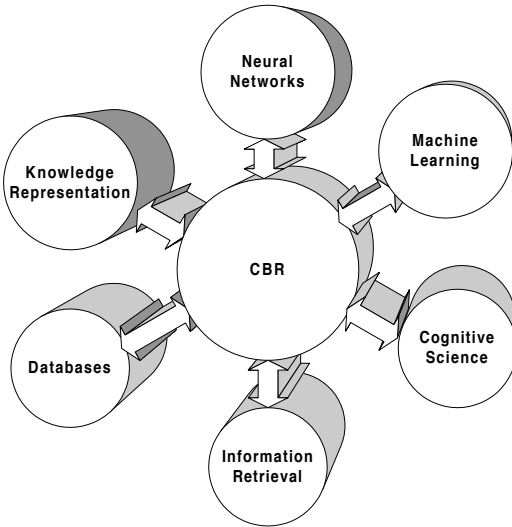
Because of its integrative nature, it is not surprising that most publications that cover the topic of CBR extensively, contain a chapter (or a section) concerning the relations between CBR and various other disciplines.

The previous chapters focused on the use of CBR for specific applications. In this very last chapter, we will step back and take a look at the relations between CBR and various neighboring disciplines from an application-independent point of view.

Obviously, we have to restrict ourselves to a selection of related areas. Our focus is on aspects concerning the relations to other fields in computer science and artificial intelligence, the relationship to cognitive science must be neglected, but is covered excellently in other publications (e.g. Kolodner 1993).

For each of the related areas we consider, we are especially interested in the following two aspects:

- Uses of techniques from related areas within CBR.
- Influences of CBR in related areas.



**Fig. 13.1.** CBR and (selected) related areas

Even within computer science and artificial intelligence, we have to select some areas, and are not able to give an exhaustive overview. The areas of interest are described below.

Due to the explosive growth of the Internet and especially WWW, the topic of *intelligent retrieval of information* has made it to the front-pages of daily newspapers and caught the eye of a very broad audience. Web-based intranets that allow the content-oriented retrieval of information are predicted to become the backbone of corporate-wide information systems. Simultaneously, they are used to establish the presence of the company on the Internet serving as the interface to its customers.

Since intelligent retrieval is one of the key aspects of CBR, there is a big window of opportunity for case-based reasoning in this area. Hence, within computer science we focus on areas related to this topic. Obviously, *information retrieval* is of great interest. *Databases*, especially newer tendencies such as *vague retrieval* and *multi-dimensional access methods*, is the second area whose relationship to CBR is presented.

We also look at *knowledge representation* techniques from the perspective of intelligent retrieval. In this area, we focus on *description logics*. Description logics are a formalism from artificial intelligence, that, due to their formal semantics and their powerful inferences, have caught the interest of researchers in case-based reasoning as well as information retrieval and databases.

Another area from artificial intelligence we investigate is *machine learning*. Here, we focus on *case-based learning*. After presenting a formal framework for case-based learning the topics of representability and learnability within that framework are discussed.

## 13.2 CBR and Information Retrieval

Although case-based reasoning (Aamodt and Plaza 1994; Kolodner 1993; Slade 1991) and Information Retrieval (IR) (Rijsbergen 1979; Salton and McGill 1983; Frakes and Baeza-Yates 1992; Fuhr 1993b; Schäuble 1993) have a great deal in common and are often used for similar tasks, there are relatively few systems and publications dedicated to the integration of the two techniques. The first scientific event focusing on this topic was the 1993 AAAI Spring Symposium on case-based reasoning and information retrieval (CBR-AAAI-WS 1993). This symposium had the goal to explore the opportunities for technology sharing. Since then, a number of activities between the two fields have been started, e.g., the invited talks of noted IR researchers at CBR Workshops and conferences.

### 13.2.1 Similarities and Differences – Common Perceptions

**Similarities.** Most people agree that CBR and IR pursue the same basic approach, and can be regarded to follow similar goals. The main similarities are:

*Retrieval of relevant information.* CBR and IR both focus on retrieving *relevant* information from a set of collected data.

*Easy database querying.* Both allow easy and flexible database querying.

*Approximating relevance by similarity.* Both fields approximate the relevance of the data entities w.r.t. the query by computing the similarity of the respective entities to the query. Hence, they deliver a collection of *inexact matches* as the retrieval result.

**Differences.** Nevertheless, CBR and IR are regarded as two different fields by the vast majority of people. This viewpoint is often substantiated by setting out the following major differences between CBR and IR:

*Data Type.* CBR and IR operate over different data types. Whereas IR methods mainly operate on textual data, traditional CBR methods operate on vectors of several basic data types, mainly numbers, intervals, dates, symbols and strings. Recently, CBR researchers also addressed textual data as shown in Chapter 5.

*Amount of Data.* IR methods can handle amounts of data much larger than those manageable by CBR. CBR Systems often operate on a few hundred cases. Systems capable of handling some thousand cases are the exception. In contrast to this, IR can search hundreds of thousands, and up to millions of documents, which consume gigabytes of memory.

*Use of knowledge.* IR systems operate without using knowledge about the user's problem solving task. They provide a generic indexing and retrieval engine operating on textual data. This generic, knowledge agnostic method, can be used for a wide scope of tasks, but has its limitations

in accuracy of the retrieval with respect to the queries. CBR systems use knowledge of the problem-solving process to build effective case indices to improve the accuracy.

*Evaluation Techniques.* The information retrieval community has a strong tradition of evaluating its techniques. There is a well developed experimental methodology, based on standardized test collections and evaluation criteria such as precision and recall (Rijsbergen 1979). New proposed methods are evaluated with this methodology this, in CBR every system works with its own data, new methods and procedures are tested with this specific data. Therefore, a comparison of different systems is difficult.

*Adaption.* IR is only concerned about the retrieval of information, whereas a major part of case-based reasoning is the adaption of the retrieved cases. However, the bigger part of the commercially available case-based systems do not employ any adaption of the retrieved cases.

### 13.2.2 Current Forms of Integration

At present, mainly two forms of integration between CBR and IR can be found. The first one more or less perpetuates the distinctions from above. The second form of integration is a first attempt to overcome these distinctions and limitations.

**Coarse Integration – CBR for IR focusing.** One approach is a coarse integration of CBR and IR, where CBR is used to drive IR. Not surprisingly, it is proposed in the areas where the user must have access to large corpora of texts in addition to a set of cases; help-desk systems (Barletta 1993) and law (Rissland and Daniels 1995; Daniels and Rissland 1997). In this scenario, there are different CBR and IR modules dedicated to, for example, the technical documentation and the diagnostic episodes. One suggested way of using the system would be that a user interacts first with the CBR module in order to come up with a good idea and starting point for a search within the IR system. Hence, the information provided by the case-based reasoner is used to formulate a better query to the IR system and to provide better overall results.

**Fine Integration – Providing a text data type in CBR systems.** Often a finer integration of CBR and IR is needed. In nearly every application area, some kind of textual attributes of the cases have to be represented. The best way to do so is to provide an additional data type for texts (Kamp 1993). This data type extends the set of data types normally provided by CBR systems, such as symbol sets and numbers; that is, it provides the mechanisms developed in IR for free-form text retrieval, e.g., *n-grams* or *inverted files*<sup>1</sup>.

---

<sup>1</sup> We cannot go into details concerning these IR techniques, and refer the reader to the respective IR literature, e.g., Rijsbergen 1979; Frakes and Baeza-Yates 1992, etc. Chapter 5 also contains a brief discussion of these techniques.

The retrieval functions of CBR then operate on this basic data type in the same way as they operate on numbers or intervals. Obviously this technique can be extended to other data types now handled by IR techniques, such as audio and images.

### 13.2.3 Recent Developments - IR and CBR come closer

The latter type of integration is a first indication of the fact that the distinction between the two fields is not that strict as the differences set out in Section 13.2.1 may suggest. It is an indicator for a process that brings the two fields closer together. Over the last couple of years, the way people (not only scientists) access and search for information has been revolutionized. Information is at the fingertips of everyone, the people suffer now from information overload instead of having problems getting access to the information. In sync with this revolution, both the information itself and the interaction model have changed and pose new challenges to IR and CBR:

*Structured multimedia documents.* Most textual documents nowadays are written on computers, using text processors and stored in descriptive formats such as RTF, L<sup>A</sup>T<sub>E</sub>X, SGML and HTML. Hence, a description of the structure is already contained in the original document, it is at least easily accessible. In addition, these documents contain pictures, graphs etc. Web pages take this multimedia aspect to the extreme but are in no way special.

*Automatically generated documents.* More and more documents are generated automatically from databases. Moreover, these documents are often generated on demand as the result to queries to this databases.

*Interactive Systems.* Users want to be in control of the retrieval process. Retrieval is no longer a process of submitting a query, waiting, and collecting the results. Users want to refine their query on the fly, follow hyperlinks within the retrieved documents, etc.

*Semantic Retrieval.* The techniques developed so far are often not sufficient to fulfill the needs of today's users, especially experts. They want retrieval by content using the available knowledge, not only retrieval based on syntactic criteria such as *n*-grams and inverted files.

IR and CBR have reacted to these new challenges in a number of ways that actually bring the two fields closer together:

**IR systems operate on other data types than text.** Whereas textual data is the natural data type of information retrieval, several systems support other data types such as *numbers*. Additionally the techniques developed for text retrieval were successfully transformed to handle multimedia data types such as *sound* and *pictures* (Glavitsch and Schäuble 1992; Glavitsch et al. 1994; Mittendorf et al. 1995). Another data type developed within IR are *proper names* (Pfeifer et al. 1995b). Here, techniques like *Soundex* and Phonix cope with the problem of finding similar sounding names.

**IR systems operate on semi-structured and structured data.** To accommodate the change in the original document format towards structured documents, more and more IR systems are operating on semi-structured and structured documents (e.g., Shoens et al. 1993; Wilkinson 1994). One area that traditionally operates on semi-structured data is that of bibliographic retrieval. There are two variants of systems operating on structured data.

1. A number of systems, such as freeWais-sf (Pfeifer et al. 1995a) and HARVEST (Brown et al. 1995) first parse textual documents, or better, documents available in some markup language like SGML or HTML in order to gather the structural information. Then, these systems index the documents w.r.t. the gathered structural information. These kind of systems most often are restricted to simple attribute-value vector representations of the document, allowing only for textual, and maybe numerical attributes.
2. On the other hand there is the sub-area of fact retrieval that integrates IR techniques with databases (Motro 1988; Fuhr 1990; Fuhr 1992). These systems more or less operate directly on the structures of the database.

**IR and CBR systems make use of available knowledge.** In IR as well as in CBR there is rising interest in using knowledge to enhance the retrieval result. Traditionally, there is the subfield of “intelligent IR” that employs AI techniques for IR. This interest in knowledge-intensive methods was boosted in the late 80’s by van Rijsbergen and his introduction of the logical model of IR (Rijsbergen 1986; Chiaramella and Chevallet 1992; Sebastiani 1995). In this model, the relevance of the documents w.r.t. to a query is determined as an implication  $d \rightarrow q$  in some logic. Hence, the transition from “traditional IR” to “intelligent IR” can be characterized as the transition from index terms to concepts and from matching to inference:

*“... to design the next generation of IR systems, we will need to have a formal semantics for documents and queries. This semantic representation will interact with other types of knowledge in a controlled way, and this way is inference!”* (van Rijsbergen, SIGIR89)

Right now, knowledge is employed in various forms in IR. Thesauri (esp. the WordNet thesaurus) are used as simple semantic nets (Kazman and Kominek 1997). Conceptual graphs (Kheirbek and Chiaramella 1995), (Probabilistic) Datalog (Fuhr 1995a; Fuhr 1995b) as well as frames and scripts have all been and are used within IR.

The logical model of IR is seen by a large fraction of IR researchers as one of the main themes within future IR research<sup>2</sup>, especially in the area of multimedia information retrieval. Besides Datalog and probabilistic extensions of it (Fuhr 1995b), several other logic formalisms are considered, such

---

<sup>2</sup> At least that was the consensus at the 1995 European Summer School on Information Retrieval.

as the Dempster-Shafer theory of evidence in conjunction with situation theory (Lalmas 1997), and logical imaging (Crestani and Rijsbergen 1995), a concept based on conditional logic is another approach.

However, the most active approach to incorporate knowledge into the retrieval process are description logics (Fuhr 1993b; Meghini et al. 1993; Sebastiani 1994; Sebastiani 1995; FERMI 1996). The previous chapters have shown that there is an increased interest in CBR on object-oriented case representations and other knowledge-intensive methods of CBR. Not surprisingly, description logics are also the central approach to incorporate knowledge within CBR. We will have a closer look at description logics in the Section 13.4.

**IR and CBR actively integrate techniques developed in the other area.** We have already seen how IR techniques are integrated into existing CBR systems. However, CBR techniques are also used within IR systems. For example, CBR is used for the guidance of user sessions in (Tißen 1991; Tien 1993). In these systems, old sessions of the user interacting with the retrieval system are stored as cases and used to guide the current retrieval session.

**CBR systems operate on larger collections.** Whereas older CBR systems often relied on linear search for finding the best cases in the case library, technologies developed recently allow for much larger case-bases. For example, Wess (1995) reports a case-base with 16,000 cases realized with INRECA, Case Retrieval Nets (cf. Section 3.4) are used by Lenz and Burkhard (1996a) to provide access to the last minute offers – approx. 200,000 cases – of a major German tour operator. More recently, Case Retrieval Nets have been used to directly operate on textual documents by parsing these documents for certain keywords and interpreting these as information entities of the Case Retrieval Net (cf. Section 5.7.1, Lenz and Burkhard 1997b). Kitano et al. (1992), Shimazu et al. (1993), and Kitano and Shimazu (1996) realize case-based systems on top of corporate-wide data-bases.

**Towards an evaluation methodology for interactive IR systems.** Our last observation is a negative one. With the advent of interactive IR systems the evaluation of IR systems also has to cope with several methodological problems. Recall and precision measures lose some of their importance in advanced IR systems that provide links between information units and which offer navigation and browsing facilities. In such systems, a query might just serve to find some suitable starting points for exploring the information space in search for useful information. Moreover, the link between traditional measures of effectiveness and optimal retrieval is not clear in these contexts and will have to be re-examined. IR has reacted by starting the ESPRIT project MIRA to come up with an evaluation methodology that takes these changes into account.

### 13.2.4 Summary

CBR and IR are often considered as two areas of computer science that share some basic ideas and goals, such as the retrieval of relevant information, but use different technologies to achieve these goals. We claim that, although this is true as long as standard (or should we say legacy) systems are considered, in the long run, both fields will more and more move towards each other. As long as only the retrieval step is considered, the borders between both areas will be blurred and there will be a large overlap in the intelligent retrieval techniques that are used in both fields. In our opinion, in a few years, labeling a certain system or method as a CBR or an IR system will be largely a question of preference than of technical differences between the two fields.

## 13.3 CBR and Databases

Generally, databases and case-based reasoning are differentiated in the following way:

*“Database systems are designed to do exact matching between queries and stored information, while the goal of CBR is to retrieve a ‘most similar’ case or set of similar cases. The most similar cases may include conflicts with some of the attributes that were specified in the retrieval query. In CBR, whether a particular case should be retrieved depends not only on the case itself, but whether there are better competitors.”*  
 (Leake 1996, Chapter 1)

However, similarly to the relation between CBR and IR, this difference is only true as long as “traditional” relational database management systems are considered. Moreover, this comparison focuses on the retrieval result and neglects other main differences between the two fields. A number of differences, such as transactions, recovery, etc., can be ignored for the moment, but the following two aspects are relevant for the retrieval:

- Firstly, databases operate on data sets that are orders of magnitude larger than the case bases of CBR. Hence, the retrieval and access methods include the management of secondary storage.
- Secondly, the access methods are fully dynamic, i.e., insertion and deletion of data items can be arbitrarily intermixed with queries, without performance loss.

CBR today mostly ignores these topics, but they are of paramount importance if larger case bases should be supported. In this section, we first briefly review current forms of integration between the two fields. We then have a look at two sub-areas of current database research that are concerned with “non-standard” databases and are closely related to case-based reasoning.

Firstly, we present different approaches for handling imprecise and vague information in databases. Secondly, we review multi-dimensional access methods that were developed within database research over the last couple of years. These access methods pay attention to the above aspects and, as we will see, are very well suited to implement CBR systems. Up to now, CBR has paid very little attention to these access models, only *kd*-trees have been considered (Wess et al. 1993a; Wess 1995).

### 13.3.1 Using Databases as a CBR Backend

The most prominent current use of database techniques within CBR is to use database management systems (DBMS)<sup>3</sup> as a backend for the CBR system. Two different forms can be distinguished (see Wess 1995):

*DBMS as a persistent case store.* Here, databases are only used as a persistent store for the cases. The cases are stored within a database, they are bulk loaded while starting the CBR system. The index structures are computed and stored in main memory, they contain only pointers to the cases stored in the database. Throughout the CBR session, these index structures are used for the retrieval and, if necessary, the original cases are retrieved from the database.

Actually, the CBR-ANSWERS system described in Section 5.5 makes use of a database this way.

*Realizing a CBR system with database means.* In this scenario, the database system plays an active role. The CBR system is realized with database means. If relational systems are used, the similarity based retrieval is performed by generating tables encoding the similarity measures as well as the appropriate SQL-queries (Kitano et al. 1992; Shimazu et al. 1993; Kitano and Shimazu 1996). The difficulties then are the generation of adequate queries and the integration of this query generation within a typical CBR architecture.

The situation is different when object-oriented database management systems (OODMBS) are used. Since most object-oriented databases are lacking a descriptive query language, they are tightly coupled with a programming language or they provide their own. Ironically, this conceptual deficit of OODBMS makes it easier to realize CBR systems, since the retrieval process can be directly realized in the programming language of the database. An example of this approach has been given by Öchsner and Wess (1992).

### 13.3.2 Incompleteness/Vagueness in Databases

Handling of incomplete and vague data is closely intertwined with case-based reasoning. However, also within database research there is a growing interest

<sup>3</sup> Most often relational database systems (RDBMS), but also object-oriented ones (OODBMS) are used in newer systems.

in handling of incomplete and vague data. This research originated from the topic of missing attribute values in relational database management systems. An excellent overview of the various methods to cope with incompleteness and vagueness in data bases, as well as artificial intelligence, is given in Parsons (1996). Due to space constraints, we are only able to note some interesting facts:

*Use of Dempster-Shafer Theory.* A number of database researches suggested to use the Dempster-Shafer theory of evidence in order to cope with this problem (e.g. Lee 1992; Bell et al. 1996). This is especially interesting since there is a general understanding in artificial intelligence that Dempster's rule is computationally too expensive (Orponen 1990). However, it is argued that it is only exponentially complex in the worst case, and linear algorithms for special cases are devised. The result of Richter (1994) concerning the relation between the Hamming distance and Dempster-Shafer theory is, therefore, an interesting link between the two fields.

*Probabilistic Datalog.* Another interesting approach is to use a probabilistic version of Datalog (Fuhr 1990; Fuhr 1993a; Fuhr 1995b; Fuhr 1995a) for modeling the retrieval of incomplete data. This technique has been developed in the area of information retrieval, an indicator that all three areas are moving into one direction as far as the intelligent retrieval of information is concerned.

*Approaches based on similarity measures.* The approach most similar to traditional CBR techniques is the one pursued in Motro (1988). It presents an extension to the relational data model by so-called *data metrics*. These data metrics are nothing else than distance measures defined over the various attributes, combined by using a weighted Euclidean distance. The retrieval itself delivers a ranking on the retrieved tuples. All in all, this can be classified as a classical CBR approach.

### 13.3.3 Multi-dimensional Access Methods

Traditional database management systems (DBMS) normally use some variation of the B<sup>+</sup>-tree for single-attribute indexing. However, today various applications need to deal with multi-attribute, i.e., multidimensional, data; Geographic information systems (GIS) operating on spatial objects (either 2D or 3D) is the oldest and most established application area. Newer application areas include multimedia databases (images, audio, video), time series, string and DNA databases, as well as data mining and online analytical processing (OLAP). These application areas typically operate on higher dimensional data (typically  $10 < d < 100$ )<sup>4</sup>. To efficiently handle multi-attribute data, one needs multidimensional access methods.

<sup>4</sup> Within this picture, information retrieval tasks can be (and are) modeled as string databases with a typical dimensionality  $10^4 < d < 10^6$ . E.g., trigrams over a 27 character alphabet (26 case-insensitive alphabetic chars plus 'other')

The main problem in designing multi-dimensional access methods is that there exists no total ordering among spatial objects that preserves spatial proximity. Hence there is no easy and obvious way to extend the well understood and efficient one-dimensional access methods from traditional databases in order to handle multidimensional data.

Over the last two decades, research in spatial and multi-dimensional database systems has resulted in a wealth of access methods. In this section, we investigate to what extent these access methods can be and have been used for case-based reasoning. Due to space limitations, we can only give a very brief introduction to the various access models. We refer the reader to the excellent overviews of Gaede and Günther (1996) and Güting (1994) for an in depth analysis of the various methods.

**Query types.** We first present the scope of queries that is typically supported by multidimensional access methods:

*Exact match queries.* Given a query object find all the objects in the database with the same geometry.

*Range queries.* There are several variants of range queries. The *window query* finds all objects in the database that share at least one point with the query object, a  $d$ -dimensional interval. The *point query* is a specialization of this query type, where the query interval is reduced to a point in  $d$ -dimensional space. Additional variants are the *enclosure query*, the *containment query* and the *intersection query*. These operate on arbitrary input regions as query items and return the objects that are either fully enclosing, are fully contained, or have a non-empty intersection with the query item.

*Nearest-neighbor queries.* These are the well known nearest neighbor queries that, given a query item, find the item that is closest or most similar to the query item. Variants are *k-nearest neighbor queries* that find the  $k$  elements closest to a query, and *fixed-radius-neighbor queries* that retrieve all objects with a distance of at most  $r$  from the query item.

Obviously, all these query types can be used for case-based reasoning. Most prominently, the nearest-neighbor queries can be directly used to determine the most similar cases. However, there are also uses of the range and exact match queries, e.g. in the preselection step of a case-based reasoner. These are also of use when expert users want to be in control of the retrieval (Kamp et al. 1996).

The different multidimensional access methods vary in the query and object types they support. In particular, one has to distinguish access models that operate only on point data and access models operating on data with spatial extent.

---

results in  $d = 19683$ , vector-space models based on terms typically are in the range  $100000 < d < 500000$ .

**Point access methods.** The most prominent access method for point data is the *kd*-tree, developed by Bentley in the 1970s (Bentley 1975; Friedman et al. 1977; Bentley 1979). It is more or less a straightforward generalization of a binary decision tree to  $d$  dimensions. Most importantly, algorithms for nearest-neighbor search were devised (Fukunaga and Narendra 1975; Friedman et al. 1977). In principle, it is a quite static data structure working on internal memory, so it is not optimal with respect to these two important aspects. Later, various extensions, such as the *adaptive kd*-tree (Bentley and Friedman 1979) and the *k-d-b-tree* (Robinson 1981), an extension of the *kd*-tree that operates on secondary memory, were proposed.

The basic *kd*-tree was adopted in case-based reasoning (Öchsner and Wess 1992; Wess et al. 1993a; Goos 1994) and further optimized for the needs of CBR by Wess (1995). Optimizations include dynamic bounding boxes for speeding up nearest-neighbor queries and the extension to unordered attributes (see also the description of the INRECA project in Chapter 3). However, as far as we know, neither one of the *kd*-tree extensions, nor any of the following access methods has been considered for use within CBR.

Other tree based access models for point data include the *hB-Tree* and its extension the *hB<sup>II</sup>-Tree* (Evangelidis et al. 1997), as well as the *LSD-Tree* (Henrich et al. 1989). The latter is interesting because it is an extension of the adaptive *kd*-tree that used a special split strategy and balances the external path length using a clever paging algorithm.

It has been noted that most of the multidimensional access models were originally spatial access models, designed for two or three dimensional data (Lin et al. 1994). In principle, they can be scaled to arbitrary dimensions, but there is a severe performance loss with most of the models. Since CBR normally deals with problems having a dimensionality between 10 and a few hundred, the question arises if the generalized spatial access models are really best of breed. Most notably, the nearest-neighbor search algorithm given by Friedman et al. (1977) does almost as much work as linear scanning for  $d > 9$ . The *TV-Tree* presented in Lin et al. (1994) is designed to work with high-dimensionality data, using varying length feature vectors. This method most definitely is of interest for CBR.

A second class of point-based access models is based on hashing methods rather than tree structures. Prominent examples are the *Grid-File* (Nievergelt et al. 1984) and the *BANG-File* (Freeston 1987).

**Spatial access models.** It is often argued, or implicitly assumed, that cases are points in  $d$ -dimensional space and the retrieval process is built upon this assumption. In our experience, this assumption is wrong. Most often, cases are incompletely described; the values of certain attributes are unknown, or only vague.

CBR systems try to cope with this problem by only considering known attributes, introducing a special value *unknown*, or using qualitative scales. All these approaches have certain flaws. For example, in Wess (1995) *unknown*

is introduced as a special value and hence can be used for attributes. However, it is impossible to restrict the values of an attribute to be in a certain interval. Moreover, the indexing structure is not symmetric. Since the developed architecture was devised for classification tasks, the classification result is always unknown in the query item. Hence, the classification attribute is not used when building then index structure, it is treated specially. It is not possible to use the index structure for answering queries of the form: “All the cases having a certain classification  $x$ ”.

Another way to cope with this problem is to model cases as objects with a spatial extent. Hence, multidimensional access methods for objects with spatial extent are of interest for CBR. Most of the access models so far approximate spatial objects using their minimum bounding rectangles, i.e.  $d$ -dimensional intervals.

Using this approximation, spatial objects in  $d$  dimensions can be represented as points in  $2d$ -space and then the point access methods from above can be used. Unfortunately, this approach has several disadvantages. Firstly, the formulation of point and range queries is usually much more complicated than in the original space. Secondly, the distribution in dual space may be highly non-uniform, even though the original data is uniformly distributed. Thirdly, images of objects that are close in the original space may be arbitrarily far apart in dual space. Fourthly, and most important, more complex queries, such as distance based queries, may not be expressible at all.

On the other hand, there are various multidimensional access methods that operate directly on the spatial data. The most prominent ones are the *R-Tree* (Guttman 1984) and its variants. However, spatial extensions of the *kd-tree*, such as the *extended kd-tree* and the *SKD-Tree*, are also proposed. The latter contains mechanism similar to the dynamic bounding boxes proposed by Wess (1995) in his extensions for CBR.

All these access models still operate with ortho-rectangular objects. However, often intervals are not a good approximation of the data objects enclosed<sup>5</sup>. Polyhedral Tree Structures like the *Cell Tree* (Günter 1989) and the *BV-Tree* (Freeston 1995; Freeston 1996) facilitate searches on data objects with arbitrary shapes, especially polyhedra.

### 13.3.4 Integration via Extensible Databases

Until recently, all these multidimensional access models were mainly of academic interest, only few of them were deployed in commercial database servers. However, the world has radically changed in the last years. Due to applications like inter- and intranets, OLAP, etc., most database vendors are working on extensible database servers right now. These so called object-relational database management systems (ORDBMS) can be extended

---

<sup>5</sup> See Kamp (1996) for an example from case-based diagnosis where this is the case.

to incorporate multidimensional data types like spatial information, texts, etc. Informix and Oracle are already providing such universal servers. Multi-dimensional access structures for geographic information systems as well as for multimedia data and texts are already available. Up to now, there are no extension modules (e.g., *data blades*, *data cartridges*, or whatever the database vendor calls them) available for doing case-based retrieval.

We think that this is a big, if not the greatest, opportunity for the developers of CBR systems. However, in order to do so successfully commercially<sup>6</sup>, a lot of research lies ahead.

Firstly, and most important, the case-based reasoning techniques have to be scaled up to handle large case-bases that are stored on secondary memory. One way to do so, is to host CBR techniques on top of multi-dimensional access structures. Hence, the various multi-dimensional access methods have to be carefully analyzed w.r.t. their relevance and usefulness for CBR and the methods must be chosen. The presentation of the various multi-dimensional access methods in the previous section only gives some hints. The following questions must be answered in detail:

- Does the access method support the dimensionality of CBR?
- Does the access method support the right kind of data? What are the implicit assumptions on the data type?
- How can the supported query types be used for CBR?
- Can missing query types, e.g. nearest-neighbor, be integrated/implemented? If so, how efficient?

The existing access methods must then gradually evolve to access methods optimized for doing case-based retrieval.

Secondly, solutions must be implemented and tested on real-world data. Evaluation methods must be devised. Standard benchmarks (both for testing the quality of the retrieval as well as the performance) must be agreed upon. This is an area where CBR lags behind information retrieval and databases.

### 13.3.5 Summary

Besides hosting CBR technology on top of relational databases, as is done today, there are two areas within database research that are relevant to CBR. Firstly, there are several approaches to handle incomplete and vague data within data bases that should be worth looking at. One interesting thing to note is the use of the Dempster-Shafer Theory of evidence within these approaches, as well as in some approaches to information retrieval. Secondly (and this is in our opinion the most promising area for a tight and fruitful integration of CBR techniques with databases) there are multi-dimensional access models. A wealth of access models have been defined, all of which may

---

<sup>6</sup> At least in the long run. It is, of course, possible to do a quick port of some existing technology, but this will be suboptimal.

be applicable for CBR. Up to now, only *kd*-trees have been investigated and used by CBR. However, since their introduction twenty years ago, a lot of progress has been made, and may be of interest to CBR.

## 13.4 CBR and Knowledge Representation

As we have seen in Section 13.2.3, there is a trend in IR as well as in CBR to integrate more knowledge into the systems. Most of the current state-of-the-art CBR systems, at least the non-commercial ones, are object-oriented and allow structural domain knowledge to be represented.

We have also seen that description logics play a major role in IR and CBR when the formal semantics of the retrieval and automatic semantic indexing are considered. On the other hand, description logics<sup>7</sup> are a knowledge representation scheme that combines strict semantic and powerful inferences with an object-oriented representation of the knowledge (Brachman and Schmolze 1985). Therefore, description logics can be viewed as a powerful extension of object-oriented representations.

Hence, it is not surprising that description logics are the major topic of research w.r.t. the integration of object-oriented databases and knowledge representation schemes (see for example (Beneventano and Bergamaschi 1997) and the various proceedings of the KRDB workshops (Baader et al. 1995; Baader et al. 1996)) and one of the major topics in knowledge representation in general.

Within case-based reasoning, description logics are used for various applications including law (Ashley and Alevan 1993), planning (Köhler 1994; Napoli et al. 1996), service support (Kamp 1994; Kamp et al. 1996; Kamp 1996), software reuse (Fernandez-Chamizo et al. 1996), and retrieval of bibliographic data (Kamp 1997). However, most of the developed techniques are independent of the application area. In this section, we briefly discuss the reasons for using description logics in CBR, present the basics of description logics and sketch their use for intelligent, similarity-based retrieval.

### 13.4.1 Description Logics

Description logics (DL) have a long tradition (originating from the KL-ONE system, Brachman and Schmolze 1985) in organizing information with a powerful representation scheme, clearly defined model-theoretic semantics and powerful inferences based on that semantic. In short, they can be used to overcome the following shortcomings of object-oriented representations:

---

<sup>7</sup> Finally, the researchers working in the field have settled on the term *description logics*. Earlier, description logics were known as *terminological logics*, *KL-ONE like languages*, etc.

*No seamless integration of knowledge.* Whereas it is possible to integrate device knowledge into object-oriented systems, it is not clear how the integration of physical laws and models of behavior could be done in a uniform way.

*No declarative semantics.* In a number of application areas, especially when the users are experts, the numerical similarity values delivered by most case-based systems have too little explanatory value; their semantics are not transparent to the users.

*No automatic semantic indexing.* Whereas the syntactical indexing of cases (e.g., assigning a bucket in a *kd*-tree, Wess 1995) could be done automatically; a semantic index, such as the membership of an object to a certain class, must be assigned by the user (Kamp 1993).

**Basics.** Description logics consist of two parts; a so called Terminological Box (TBox) containing terminological knowledge and an Assertional Box (ABox) containing assertional knowledge. Since we can not elaborate on the basics of description logics, we only give a brief introduction in the following and refer the reader to the relevant literature (e.g. Baader et al. 1992) for a more detailed introduction:

**Definition 13.4.1 (TBox).** *Concept terms allow for a structured (object-oriented) representation of a relatively large fragment of first order-logic. Starting with primitive concept and role terms, new concept terms can be constructed from others by a set of concept forming operators. There are mainly three categories of such operators (Hanschke 1996):*

- Boolean operators (e.g. (and  $C_1 \dots$ ), (or  $C_1 \dots$ ), (not  $C_1$ ))
- Role Forming operators (e.g. composition of roles (compose  $r_1 r_2$ )).
- Operators on Role Fillers (e.g. quantification (some  $r C$ ), (exists  $r C$ )).

Terminological axioms in the form (define-concept  $CN C$ ) associate a concept name  $CN$  with a concept term  $C$  and are used to define the relevant concepts of an application. Terminological axioms are therefore an extension to the class definitions of an object-oriented CBR approach. A TBox ( $\mathcal{T}$ ) is a finite set of terminological axioms.  $\square$

**Definition 13.4.2 (ABox).** *Concrete objects are instances (individuals, objects) of concepts. A new instance  $o$  can be introduced into the ABox via (define-distinct-individual  $o$ ), and assertions  $\alpha$  concerning the membership of an instance  $o$  to a concept  $C$ , or about existing relations (functions) between two objects  $o_1$  and  $o_2$  can be made through (state (instance  $o C$ )) or (state (related  $o_1 o_2 r$ )), respectively. The set of assertions constitutes the ABox  $\mathcal{A}$ .*

$\square$

What distinguishes a description logic approach to CBR from pure object-oriented ones, is that one is able to formally define a declarative model-theoretic semantics for the T- and ABox constructs by means of an interpretation function  $\mathcal{I}$ , e.g. (and  $C_1 C_2$ ) $\mathcal{I} = C_1^{\mathcal{I}} \cap C_2^{\mathcal{I}}$ . These semantics allow

a formal definition of a number of powerful inferences to be given, providing “intelligent” (semantic) access to information, that is independent of the syntax.

**Definition 13.4.3 (Inferences).** *In our context, the following inference services are of particular interest:*

- **Consistency Test.** *The consistency test checks if an ABox (w.r.t. a TBox) is consistent (i.e. it has a model):  $\mathcal{A}^{\mathcal{T}} \neq \emptyset$ .*
- **Concept Classification.** *Concept Classification is a TBox inference service that calculates the subsumption hierarchy, i.e., the subconcept-superconcept relationships between concepts. A concept  $C_1$  subsumes another concept  $C_2$  iff. for  $C_2$  there exists also a model for  $C_1$  (i.e.  $C_1^{\mathcal{T}} \supseteq C_2^{\mathcal{T}}$ ).*
- **Object Classification.** *Object classification is an ABox inference service that determines, given an object  $o$  of the ABox, the set of most specific concepts  $\overline{C}(o) = \{C_1, \dots, C_n\}$  that this object belongs to, i.e.  $\overline{C}(o) = \{C_i \in \mathcal{T} \mid \{\mathcal{A} \cup \neg(o : C_i)\}^{\mathcal{T}} = \emptyset \wedge C_i \text{ minimal in } \mathcal{T} \text{ w.r.t. subsumption}\}$ .*
- **Weak object classification.** *Weak object classification, in contrast, delivers the set  $\underline{C}(o)$  of the most general concepts the object may be specialized to if new information is added, i.e.,  $\underline{C}(o) = \{C_i \in \mathcal{T} \mid \{\mathcal{A} \cup (o : C_i)\}^{\mathcal{T}} \neq \emptyset \wedge C_i \text{ maximal in } \mathcal{T} \text{ w.r.t. subsumption}\}$ .*
- **Retrieval.** *Retrieval is dual to the object classification problem. Given an concept term  $C$ , the set of TBox objects (instances)  $\overline{O}(C) = \{o_1, \dots, o_m\}$  that are members of  $C$ , is returned.*
- **Weak retrieval.** *Weak retrieval returns the set of objects  $\underline{O}(C) = \{o_1, \dots, o_m\}$  which are possible members of the concept.*

□

Recent work in the area of DL focused on determining the computational properties of different description languages (i.e. sets of concept terms). Since all the above inference services are defined based on the consistency test of an ABox, it is sufficient to find an algorithm for this problem. It has been shown that there exist sound and complete algorithms to determine the consistency of an ABox for a number of description languages. These algorithms are inspired by the tableaux calculus. However, only a few available systems, such as KRIS (Baader and Hollunder 1990) and TAXON, are based on this technique. Most systems (e.g., LOOM (MacGregor 1991) and CLASSIC (Patel-Schneider et al. 1991)) use incomplete structural subsumption algorithms instead; and, since they do not reduce subsumption to the consistency test of an ABox, a number of the other inferences are unavailable. This is a fact that one has to keep in mind when reading articles about CBR and description logics.

### 13.4.2 Intelligent Retrieval Based on Description Logics

All of the above inference services can be used for similarity-based retrieval:

- Object classification is the method of choice if one wants to use the system in the traditional CBR way; i.e. a new case should be solved by looking for similar cases.
- Concept classification and retrieval can be used to implement a database like service where the user poses queries against the case base and retrieves the matching elements.

**Methods based on object classification.** Object classification maintains a dynamic index of the ABox objects  $o_j \in \mathcal{A}$  with respect to the TBox concepts  $C_k \in \mathcal{T}$ . Hence, the indices (i.e., the most specific concepts a certain object belongs to) can be used as the base for all kind of similarity measures. The method proceeds as follows:

1. **Add assertions.** New assertions (e.g., parameter restrictions) about an object  $o_j$  trigger automatically a reclassification and, therefore, a reindexing of  $o_j$  and all the objects connected to it.
2. **Retrieve the indices.** Either the strong  $I'(o_j) = \overline{\mathcal{C}}(o_j)$ , or the weak variant  $I''(o_j) = \underline{\mathcal{C}}(o_j)$  of object classification, or a combination of both  $I'''(o_j) = (\overline{\mathcal{C}}(o_j), \underline{\mathcal{C}}(o_j))$  are used.
3. **Determine similar objects.** Based on the indices provided by the last step, the various well-known similarity measures can be used:
  - a) **Identical indices.** The most simple, pure relational similarity measure returns all the objects as similar that have the same set of indices as the query object. Let  $I^* \in \{I', I'', I'''\}$  and  $\aleph_M$  be the characteristic function of  $M$ . Then,  $sim_{I^*}(o_l, o_k) = \aleph_{I^*(o_l)=I^*(o_k)}(o_k)$ . Due to the inferences drawn by the classification process, even this simple similarity measure is sufficient for most purposes.
  - b) **Various association measures.** Alternatively, all the various similarity measures developed in IR and CBR (Rijsbergen 1979; Tversky 1977; Bayer et al. 1992) operating on index sets or index vectors, respectively, can be used. For example, the simple matching coefficient, the Jaccard coefficient, the Dice coefficient, the contrast rule, the cosine measure; virtually all the techniques known.
  - c) **Graph Distance between the index sets.** Moreover, since the indices, e.g. concepts are arranged in a directed acyclic graph, the structure of this graph can be taken into account, and the respective measures can be used.
4. **Display the similar objects.**

**Methods Based on Retrieval and Concept Classification.** Retrieval and concept classification are useful for a more database-like use of the case base, i.e., query answering. The inference service that is most obvious to use is the retrieval service. Methods based on it work as follows:

1. **Query formulation.** Since the query language and the concept description language are identical, an arbitrary concept term  $C_q$  can be used as query formulation.

2. Calling the retrieval inference. Compute the sets  $I'(C_q) = \overline{O}(C_q)$ ,  $I''(C_q) = \underline{Q}(C_q)$  resp.  $I'''(C_q) = (\overline{O}(C_q), \underline{Q}(C_q))$ .
3. Display the matching instances.

Alternatively, concept classification can be used, to approximate the results of the retrieval step, or better, to provide approximate retrieval. The method works as follows:

1. **Preprocessing.** For each  $C_j \in \mathcal{T}$ , call the retrieval  $O(C_j)$  and index the returned instances. Use either strong  $I'(C_j) = \overline{O}(C_j)$ , weak retrieval  $I''(C_j) = \underline{Q}(C_j)$  or a combination of both  $I'''(C_j) = (\overline{O}(C_j), \underline{Q}(C_j))$  as an index.
2. **Query formulation.** Formulate the query as a concept description.
3. **Compute the indices.** Use the concept classification  $C_q$  to determine its position in the concept hierarchy, i.e. calculate the sets of its direct upper neighbors  $\overline{C}_q = \{C_{q_1}, \dots, C_{q_n}\}$  and direct lower neighbors  $\underline{C}_q = \{C_{q_1}, \dots, C_{q_m}\}$ . Use these sets as indices.
4. **Compute the instances to be retrieved.** As in the object classification method above, use the indices as the base for computing similarity measures.
5. **Display the result.**

### 13.4.3 Incorporating Base Types

One major shortcoming of description logics is that, initially, description logics operated only on the *abstract domain*. In order to refer to concrete values of attributes, such as a force  $F$  or a name  $n$ , one needs access to base types such as numbers, symbol sets and strings. However, traditionally, this was only possible in description logic systems based on structural subsumption algorithms such as LOOM and CLASSIC. However, their integration of base types was most often ad hoc, and the semantic were unclear.

Baader and Hanschke (1991) and Hanschke (1996) have devised a general method, *concrete domains*, to incorporate base types into description logics without losing soundness and completeness of the inferences. This extension of description logics not only overcomes the deficiency of description logics w.r.t. object-oriented representations, it also allows for the inclusion of background knowledge, such as physical laws, etc. Kamp (1996) and Kamp (1997) discuss admissible concrete domains in the context of different application scenarios.

### 13.4.4 Summary

Description Logics are a knowledge representation method that has been considered in IR, CBR, and DBMS and hence, can be used to integrate aspects from all these fields. Based on a model-theoretic semantics, they provide a

powerful object-oriented representation, together with a set of powerful inferences, that allow for the automatic indexing of instances and classes. On top of these indexing services, traditional similarity measures can be implemented, but, due to the quality of the indices, often simple similarity measures, such as the simple matching coefficient, are sufficient. Concrete domains allow for the integration of primitive data types, such as numbers and strings, without losing soundness and completeness of the inferences. They can also be used to represent and use background knowledge. All in all, description logics are a viable base for building knowledge intensive case-based reasoning applications in a broad spectrum of domains.

### 13.5 Intelligent Retrieval – Summary and Outlook

The previous sections are best summarized as follows:

Conceptually, CBR, IR, and DBMS are coming closer together, meeting each other in order to provide the intelligent retrieval techniques over complex, structured data that is needed to realize the applications of tomorrow.

Integrated retrieval methods, combining relational databases, with multidimensional access structures (for geographic information systems, as well as for multimedia data) and IR methods for text retrieval are already available in extensible object-relational databases. Up to now, there are no extension modules available for doing case-based retrieval. Building such a module based on the right multidimensional access structure is, in our opinion, a great opportunity, and should be one focus of future research within CBR.

Another topic of future research within intelligent retrieval is the integration of domain and background knowledge to enhance the semantic of the retrieval. This could be done by considering and integrating techniques from knowledge representation. One approach that was pursued simultaneously and independent within IR, CBR and DB is the use of description logics, a technique that automatically computes semantic indexes of classes, as well as instances, but also is computationally very expensive. In this area, further research includes finding guidelines for finding the right tradeoff between expressiveness and complexity for different application scenarios, the search for approximations, etc.

### 13.6 Case-Based Reasoning and Learning

In the introduction of this book, the case-based reasoning paradigm is illustrated by a methodological cycle consisting of the main activities retrieve, reuse, revise, and retain. An important phase in this cycle is the retain phase

in which the system may be modified as a result of its usage and success in the past. In principal, there are the following options to adapt the system's behavior; new cases may be collected within the case base and/or the similarity concept in use may be tuned. From a certain perspective, the process of changing a system in response to its environment is *learning*. The naturalness of this approach is one of the advantages of the case-based reasoning paradigm.

There are two ways to discuss the relation between case-based reasoning and machine learning; one can analyze the strengths and weaknesses of the above mentioned approach to modify the system's behavior, subsequently called *case-based learning*; or one can suggest ways of combining case-based reasoning and classical machine learning techniques and point out benefits of such combinations. In this section, we focus our attention on the first aspect and show that case-based learning is also a quite powerful approach to improve the behavior of a case-based reasoning system. The second aspect has been addressed in Chapter 3 where the combination of CBR and induction has been explored.

### 13.6.1 A Formal Framework for Case-Based Learning

Not every change of a CBR-system is a reasonable one. Thus, there is some need to specify the intended learning behavior. In doing so, we borrow some notions from one of the most thoroughly studied areas in machine learning, namely, concept learning (cf. Dietterich and Michalski 1983; Osherson et al. 1986). Here, a learning system is considered to be a computer program that maps *evidence* on a target concept into *hypotheses* about it. Of special interest are scenarios in which the sequence of hypotheses *stabilizes* to an *accurate* and *finite* description of the target concept. Clearly, in this case, some form of learning must have taken place.

Subsequently, we discuss the following scenario: A learning system (or learning algorithm, synonymously) receives evidence about a target concept in the form of labeled examples. The labels indicate whether the corresponding example is a member or a non-member of the target concept, thereby using the labels '+' and '-', respectively. The learning system processes more and more labeled examples and generates, in every learning stage, a hypothesis. Learning systems differ in the way they build and represent their hypotheses. For instance, an ordinary learning system may generate a finite set of rules, a neural net, a decision tree, or any computer program that implements a decision function as its actual hypothesis. In contrast, a case-based learning system (case-based learner, for short) is constraint to represent its actual hypothesis in a case-based fashion; i.e., it has to output a case-based classifier consisting of a finite subset of the set of examples seen so far (the case base) and a similarity measure. We say that a learning system has successfully learned if, after only finitely many learning stages, it comes up with

a correct description of the target concept that will be maintained in every subsequent learning stage.

Clearly, it is a rather useless task to design learning systems that are able to learn exactly one concept. Consequently, one is highly interested in learning systems that are suited to learn every concept of a class of concepts. For instance, it seems to be of practical interest to have learning systems that are able to learn every concept that is expressible as a monomial, a decision list or a DNF.

In the formal framework presented above, case-based learners are just ordinary learning systems that are designed to learn every concept from a given concept class. Additionally, case-based learners are requested to output case-based classifiers as their hypotheses. Now, the following questions immediately arise. Firstly, one has to investigate representability issues, since a case-based learner can only stabilize on a correct guess provided that every concept  $c$  in the concept class is case-based representable; i.e., there exists a case-based classifier that correctly describes  $c$ . Secondly, appropriate techniques have to be elaborated that allow a case-based learner to construct corresponding case-based classifiers. Finally, it seems to be of particular interest to compare the learning capabilities of case-based learners with the capabilities of ordinary learning systems. The latter yields general insight into the pros and cons of case-based learning.

### 13.6.2 Representability

The answers to the questions posed above heavily depend on the class of concepts to be learned. In order to achieve insight that are of interest within a broader perspective, we investigate the case-based learnability and, prior to this, the case-based representability of indexable concept classes.

Let  $\mathcal{X}$  be any set, called the learning domain. Let  $\mathcal{C}$  be any subset of the power set of  $\mathcal{X}$ , and let  $c \in \mathcal{C}$ ; then we refer to  $\mathcal{C}$  and  $c$  as a concept class and a concept, respectively. A class of concepts  $\mathcal{C}$  is said to be an indexable concept classes provided there are an effective enumeration  $c_0, c_1, c_2, \dots$  of all and only the concepts in  $\mathcal{C}$  and a recursive predicate  $p$  such that, for all  $j \in \mathbb{N}$  and all  $x \in \mathcal{X}$ ,  $p(x) = 1$  if and only if  $x \in c_j$ .

For illustration, let us refer to some prominent examples of indexable concept classes. Firstly, let  $\mathcal{X}$  be the set of all strings over any finite alphabet of symbols. Then, it follows that the set of all context sensitive languages, context free languages, and regular languages form indexable concept classes. Secondly, let  $\mathcal{X}$  be the set of all  $n$ -bit boolean vectors, and let  $\mathcal{X} = \bigcup \mathcal{X}_n$  be the underlying learning domain. Then, it follows that the set of all concepts expressible as a monomial, a DNF, and a decision list form indexable concept classes.

As already mentioned, a case-based classifier is a pair consisting of a finite case base and a similarity measure. Since case-based classifiers should be used to represent concepts, some discussion of the underlying semantics is

required. Let  $\mathcal{X}$  be a learning domain, let  $CB$  be a case base<sup>8</sup>, i.e.,  $CB$  is any finite subset of  $\mathcal{X} \times \{+, -\}$ . Let  $\sigma$  be a similarity concept, i.e.,  $\sigma$  is a recursive function that assigns to any two elements  $x, y \in \mathcal{X}$  their similarity, a rational number between 0 and 1. Then it follows that the concept represented by  $(CB, \sigma)$  is the set of all  $w \in \mathcal{X}$  meeting the following requirement; there is a positive case  $(u, +)$  in  $CB$  such that  $u$  is more similar to  $w$  than every negative case  $(v, -)$  belonging to  $CB$ , i.e.,  $\sigma(u, w) > \sigma(v, w)$ . Intuitively,  $w$  belongs to the represented concept, if all of its nearest neighbors in  $CB$  have the label ‘+.’

Next, let us summarize some of the basic results concerning case-based representability of indexable concept classes. Firstly, given a (recursive) concept  $c$ , one can easily define a similarity measure that allows to represent  $c$  using a singleton case base and, thus, every indexable concept class is case-based representable. However, from a practical point of view it is more desirable to have universal similarity measures that can be used to represent any concept of a given concept class. If such a measure has been fixed, it becomes much easier to represent each concept of the class in a case-based manner. Now, it suffices to select cases that are appropriate to sit in the case base. Surprisingly, the latter approach really works. Given any indexable concept class  $\mathcal{C}$ , one can define a similarity measure  $\sigma$  guaranteeing that, for every concept  $c \in \mathcal{C}$ , there is case base  $CB_c$  such that the concept represented by  $(CB_c, \sigma)$  equals  $c$ . Moreover, it is sufficient to put at most two cases into the case base provided that  $\mathcal{X}$  is an infinite learning domain and, therefore, the concept itself or its complement is infinite.<sup>9</sup>

Finally, let us point to a problem of higher granularity. In some applications, the used similarity concept  $\sigma$  is defined via some underlying distance measure  $\Delta$  and, therefore,  $\sigma$  inherits some of the properties  $\Delta$  has. For instance, the similarity measure is necessarily symmetric<sup>10</sup> if it is defined via a metric. Theoretically, it is imaginable that similarity measures having such seemingly natural properties are less expressive. Interestingly, this phenomenon can really be observed. On the fairly concrete level of indexable concept classes, it has been shown that certain concept classes are case-based representable by non-symmetric similarity measures only (cf. Jantke and Lange (1995) for the corresponding details).

### 13.6.3 Learnability

It is known that, given any indexable concept class  $\mathcal{C}$ , one can design an ordinary learning algorithm that learns  $\mathcal{C}$  from positive and negative examples

<sup>8</sup> Naturally,  $CB$  should be conflict-free, i.e., for any  $x \in \mathcal{X}$ , it should not contain  $(x, +)$  together with  $(x, -)$ .

<sup>9</sup> In finite learning domains, the situation is much more complicated (cf. Globig and Lange (1996) for a more comprehensive discussion).

<sup>10</sup> A similarity measure  $\sigma$  is called symmetric, if  $\sigma(v, w) = \sigma(w, v)$  for all  $v, w$ .

(cf. Gold 1967). Although the algorithm proposed in Gold (1967) is completely different from a case-based learning algorithm, it proves the existence of powerful learning systems. As we will see, one can be equally successful in designing case-based learners.

Firstly, we investigate the capabilities of the most restricted type of case-based learners. Learning algorithms of this type are not allowed to modify the similarity concept in use. Thus, the similarity measure is *a priori* fixed and the learning task reduces to the task of collecting appropriate cases. Surprisingly enough, this simple approach really works, if the learner is allowed to delete cases that have been temporarily stored in the case base; this assumption usually holds (cf. Globig and Lange 1994).

In some case-based systems, as for instance in INRECA (Wess 1995; Wilke and Bergmann 1996a) and PATDEX (Althoff et al. 1990b), the used similarity concepts are also modified during their life-cycles. A comparatively simple and well known approach to model this aspect is based on the idea of assigning weights to the cases sitting in the case base (cf. Cost and Salzberg 1993). Instead of modifying the underlying similarity measure  $\sigma$  itself, the assigned weights will be tuned, only. Thereby, cases that are of higher relevance for the concept to be represented get higher weights. More specifically, a weight  $\alpha_w$  is assigned to each case  $(w, b)$  sitting in the case base. Thus, the value  $\alpha_w \cdot \sigma(w, u)$  (instead of  $\sigma(w, u)$  in the standard approach above) represents the similarity between  $(w, b)$  and a given  $u \in \mathcal{X}$ . Note that this approach may result in non-symmetric similarity concepts, even if the underlying similarity measure  $\sigma$  is symmetric. Case-based learners that implement the idea to learn by collecting cases and tuning weights are powerful in the sense that, given any indexable concept class, one can design a case-based learner of that type being able to learn the whole class (cf. Globig et al. 1997). Interestingly, these learners do not need the flexibility to delete cases from their actual case bases, in the overall learning process.

## 13.7 Summary

Having its origins in cognitive science and computer science, CBR is traditionally interdisciplinary. CBR borrows a number of techniques from other fields in AI and computer science in general; at the same time, these fields more and more use new ideas that originated in CBR.

In this chapter, we have restricted ourselves to a few selected areas related to CBR. We focused on connections to other areas in computer science and artificial intelligence, especially intelligent retrieval aspects and learning techniques. We argued that, concerning the intelligent retrieval of information, the areas of IR, Databases, Knowledge Representation, and CBR are moving in the same direction, and have already reached some kind of intersection in the area of intelligent retrieval of information from data. With respect to

learning, we have discussed under which circumstances certain concepts are learnable using a case-based learner.