An Investigation of Generalized Cases

Kerstin Maximini, Rainer Maximini, Ralph Bergmann

University of Hildesheim
Institute for Mathematics and Applied Computer Science
Data and Knowledge Management Group
31113 Hildesheim, Germany
{k_maximi|r_maximi|bergmann}@dwm.uni-hildesheim.de

Abstract. In the CBR literature from the past 25 years there is a considerable amount of research work that makes use of cases that are subspaces of some representation space rather than points in it. For cases of that kind, different terms have been used such as generalized case, prototype, schema, script, or abstract case. Our analysis of selected publications yields that on the one hand the same term is used for different concepts and on the other hand different terms are used for more or less the same concepts. So our goal is to improve the conceptual clarity by proposing an integrated classification schema for cases. We then use this schema to describe semantically founded ways for similarity definition and computation, depending on the class membership of query and case.

1 Introduction

When looking into the literature describing Case-Based Reasoning, applied to different domains and tasks, one often finds the term *generalized case* or the statement that (some) cases in the case base are *generalized*. Other publications use the concept *prototype* for a very similar application scenario, whereas in publications mostly written in the eighties *schema* and *script* are used. In newer publications the term *abstract case* and *concrete case* is introduced, which shows similarities to generalized cases.

In this paper, we argue that on the one hand the same term is used for different concepts and on the other hand different terms are used for more or less the same concept. So our goal is to improve the conceptual clarity by proposing an integrated classification schema for cases. Such a classification is needed to face the challanges of the current research project *IPQ: IP Qualification for Efficient Design Reuse*[1] funded by the German Ministry of Education and Research (BMBF) and the related European Medea project *ToolIP: Tools and Methods for IP*[2], namely how to handle parameterized cases, whose attributes are not defined independently of each other. The main problems of these generalized cases include their representation formalism, the calculation of similarity, and the implementation of a fast and efficient retrieval. The presented classification schema

[1] IPQ Project (12/2000 - 11/2003). Partners: AMD, Fraunhofer Institute for Integrated Circuits, FZI Karlsruhe, Infineon Technologies, Siemens, Sciworx, Empolis, Thomson Multi Media, TU Chemnitz, University of Hildesheim, University of Kaiserslautern, and University of Paderborn. See www.ip-qualifikation.de

[2] See toolip.fzi.de for partners and further information.

and similarity measures are not specialized to this domain; they are of general nature and can be applied to any CBR application.

The next section of this paper presents selected work from the history of CBR, using generalized cases and related concepts. We derived an integrated classification schema, proposed in section 3. Main focus of this paper lies in the analysis of the similarity measures for the different kinds of case, which is discussed in section 4. We close our paper with a summary and give ideas for future developments in this area.

2 Generalized Cases in the History of CBR

We have reviewed publications describing CBR systems in different application scenarios and different domains, e.g. systems for design, planning, and maintenance in domains like medicine, e-commerce, and architecture. We included also publications about the dynamic memory as a starting point. So the following summary bases upon selected publications of the last 25 years, dealing with the concepts of generalization and abstraction. It is in no way complete, but shows, which terms have been used over the time.

2.1 Dynamic Memory and Cognitive Model

First we want to make clear, that the idea of generalizing cases is not a radically new concept. It was already implicitly present since the very beginning of CBR and instance-based learning research [1–5].

The theory of dynamic memory [6], which is partly implemented in CBR systems [3], proposes a cognitive model and artificial methodology of a memory in which reasoning and learning are inseparable. According to this theory our general knowledge about situations is recorded in *scripts* and the memory is organized by *memory organization packets* (MOPs). MOPs have two functions: They hold general knowledge and they organize specific experiences of that general knowledge in cases. Whereby the cases are the starting point for solving problems and the general knowledge provides guidance in adapting old solutions to fit to new situations. In 1989, Kolodner and Simpson [7] introduced the *generalized episode* which is more or less equivalent to a MOP but describes a general episode instead of a specific one.

2.2 Design

In the early 90s generalization and abstraction of cases have been a topic of research in the design area. For example, every participant involved in the design of a house, like civil engineers, architects, and occupants, has another view to the design. These views are termed *different abstractions* and must be reconciled in the final design. Particularly, all constraints from each abstraction must be regarded for the final design. In the CADRE system [8] the design knowledge for each abstraction is represented in form of so called *design prototypes*, whereby a design prototype is a generalized version of a special design.

Another project in this area was the joint research project FABEL, where amongst other things DOM (domain ontology modelling) in architectural and engineering design has been developed [9–11]. The DOM system was designed to assist architects and engineers who are involved in designing the technical installation systems for highly complex buildings by usage of generic concept knowledge and episodic case knowledge. The most interesting knowledge base contains the domain knowledge and is set up with three knowledge containers: *generic knowledge*, *case knowledge*, and *scheme knowledge*. Design rules like analysis or synthesis rules are represented as *concepts* in the generic knowledge container. *Cases* are snapshots of the designs during the design process (temporally intermediate states of the layout) and therefore can be seen as episodic knowledge. *Schemas* are viewed as generalized cases, because they are created by putting together some similar cases to one more generic pattern. Thereby, equal parts are retained and different parts are replaced with variables. Schemas can be applied to many concrete situations.

2.3 Planning

The PARIS system, extensively described by Bergmann in 1996 [12, 13], reuses cases flexibly by application of abstraction and generalization. In the planing domain, the problem part of a case is a representation of the initial and finite state and the solution part is a plan that describes how the finite state can be reached by applying a sequence of operators on the initial state.

Bergmann defines three terms, namely *problem class*, *solution class*, and *generalized case*. A problem class is a set of problems which is described by a generalized description of the initial and finite state as well as a set of constraints for the initial and finite state. The solution class is a set of solutions (plans) which are described by a sequence of partly instantiated parameterized operators. The generalized case is a problem class together with a solution class. It can be reused directly for a wider range of problems than a specific case and therefors can be seen as a solution schema. The process of transformation of a specific case in a generalized case is called *case generalization*. Beside the concept of generalization also the concept of abstraction has been applied to the cases. Therefore, experience is represented at several different levels of abstraction in an abstraction hierarchy. Based on these levels *concrete cases* and *abstract cases* can be distinguished. The process of transformation of a concrete case into an abstract one is called *case abstraction*. So the PARIS system uses “abstract generalized cases”.

Another example of generalized cases in planning systems is the work of Rayner and Harris [14], who compound similar cases to form generalized cases, which are applicable to a wider range of problems.

2.4 Maintenance

In the CBR community the 4Re-cycle [15] is well established as a process description of CBR applications. For maintenance reasons the cycle has been expanded by Reinartz, Iglezakis, and Roth-Berghofer [16, 17] with two further steps, namely review and restore. For the restore step they present several operators that modify the case

base. Beside others, especially the operators *specialize case*, *generalize case*, and *abstract cases* are of interest for the scope of this paper. The specialize case operator adds a value to an unset problem attribute, whereas the generalize case operator removes a value from a problem attribute. The abstract cases operator converts two cases with an equal solution into a single case, whereby a relation determines how different attribute values are merged into a single one.

One application that uses these techniques is the ICONS system [18, 19]. It is applied in the medical domain to give advise for an appropriate antibiotics therapy for a special patient. It stores patients together with the applied therapies as cases and uses CBR technology for a fast proposal of an antibiotics therapy. Used in every-day's practice, the number of cases would increase continuously and would slow down the retrieval time. To avoid this, the case base is structured by *prototypes*, including the most frequent features of their corresponding cases. Only cases that differ significantly from their prototypes are also stored in the case base.

2.5 Product Design

Increasingly, electronics companies integrate *Intellectual Properties* (IPs) from third parties within their complex electronic systems. An IP is a design object whose major value comes from the skill of its producer [20], and a redesign of it would consume significant time. However, a designer who wants to reuse designs from the past must have a lot of experience and knowledge about existing designs, in order to be able to find candidates that are suitable for reuse in his/her specific new situation. Current CBR systems are not able to handle IPs, because they are descriptions of flexible designs and therefore spawn a design space instead of a point. Their flexibility is based on a number of parameters and dependencies, and valid value combinations for these parameters are constrained by different criteria for each IP.

Bergmann, Vollrath, and Wahlmann [21] characterize the cases that can occur and distinguish between five types: *point case*, *constant solution generalized case*, *functional solution generalized case*, *independent alternative solution generalized case*, and *dependent alternative solution generalized case*. One goal of the IPQ project is to develop case retrieval for case of all these kinds.

3 Proposed Classification Schema

3.1 Case Classification

We now introduce a formal classification schema for cases in attribute-value representation. We consider the *attribute space* $\mathbb{A} = T_1 \times \cdots \times T_n$. T_i are atomic types which are data types with single values like integer, float, boolean, symbol, string, date, or time. Each case has n attributes $A_1, \ldots, A_n$ with an associated type T_i.

We start our definitions with the easiest kind of cases, namely the *point cases*.

Definition 1. *A* point case *pc is exact one element of the attribute space,* $pc \in \mathbb{A}$*, i.e., each attribute of pc has a concrete atomic value, unequal to* $unknown$*.*

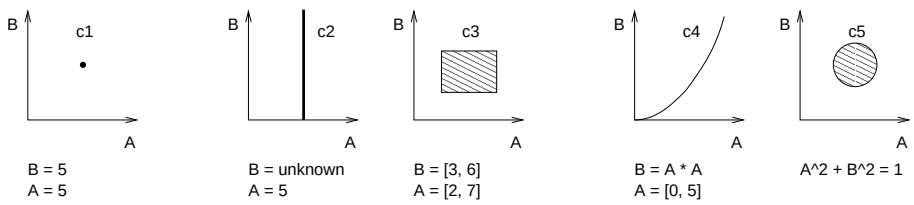

Fig. 1. Examples for the three kind of cases

A point case, e.g. c1 in figure 1, can be regarded as a special instance of another concept, namely *generalized cases*.

Definition 2. *A* generalized case *gc is an arbitrary subspace of the attribute space,* $gc \subseteq \mathbb{A}$.

Generalized cases can be interpreted as an (infinite) set of point cases. The main difference in the following two definitions is the kind how this set is defined. The most simple form of a gc is the *attribute independent generalized case* (aigc).

Definition 3. *Assume there exists sets* $\mathbb{A}_1, \ldots, \mathbb{A}_n$ *with* $\mathbb{A}_i \subseteq T_i$. *An* attribute independent generalized case *aigc is defined as* $aigc = \mathbb{A}_1 \times \cdots \times \mathbb{A}_n$. *Consequently, an aigc is a special subspace of the attribute space,* $aigc \subseteq \mathbb{A}$.

Attribute independent generalized cases often occur in applications in which not all attributes of the case are known or where attribute values have been removed, e.g. by maintenance operations. Unknown attribute values are usually interpreted as arbitrary, i.e., they can be considered to spawn the whole attribute dimension in $\mathbb{A}$. Hence, this can be represented by $\mathbb{A}_i = T_i$, e.g. c2 in figure 1. Another common reason for a case to be an aigc is because some attributes hold intervals (range) or sets as values, see c3. Further, a query is usually an aigc, because the solution attributes are not defined.

The second form of gcs are the *attribute dependent generalized cases* (adgc).

Definition 4. *A generalized case* $adgc \subseteq \mathbb{A}$ *that cannot be represented as an aigc is called* attribute dependent generalized case.

Hence, a gc is attribute dependent if the subspace it represents cannot be decomposed into independent subsets for each attribute. Dependencies can be expressed by constraints like demonstrated by c4 and c5 in figure 1. Therefore, these cases are more complex and difficult to handle during retrieval, depending on the form of the spawned subspaces. Roughly speaking, aigcs spawns a subspace which is orthogonal (except aigc with sets) and are therefore manageable in case that similarity measures are also decomposed according to the attributes. For adgcs the subspace is arbitrary and hence it may become computationally very hard to determine the similarity to a pc or another gc [21]. We will explore this issue in detail in section 4.

3.2 Abstraction versus Generalization

In recent CBR literature, the use of multiple representations at different levels of abstraction has been investigated [3, 13, 22–25, 12, 26–29]. For this kind of approach the term *hierarchical case based reasoning* [22], *stratified case based reasoning* [27] and *reasoning with abstract cases* [12] have been used so far. The basic idea behind all these approaches is to supply experience represented at several different levels of abstraction. To represent different levels of abstraction, the vocabulary is structured into several partially ordered sub-vocabularies, one for each level of abstraction. Based on these levels, two kinds of case can be distinguished:

Definition 5. *A* concrete case *is a case located at the lowest available level of abstraction, e.g. the leafs of an abstraction tree. An* abstract case *is a case represented at a higher level of abstraction, e.g. the nodes in an abstraction tree.[30]*

Concrete and abstract cases are stored in a case base for reuse when solving new problems. When a new problem must be solved, one (or several) "appropriate" concrete or abstract case is retrieved from the case base and the solution the case contains is reused to derive a solution for the current problem, e.g., by filling in the details that a retrieved case at some higher level of abstraction does not contain.

During the abstraction several cases from a level n are transformed into a single new case c_a on level $n+1$, see figure 2. The association between c_a and the abstraction

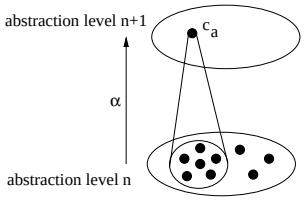

Fig. 2. Abstraction and Generalization

level n is expressed by the equation $g_n(c_a) = \{c|\alpha(c) = c_a\}$ with an abstraction function α. The abstraction function α may be a transformation to another easier to handle representation formalism or just a projection to a subset of the attributes, e.g. by removing less significant ones. Hence, α can be seen as a generalization step, because cases are converted into a gc $g_n(c_a)$ at abstraction level n. But on abstraction level $n+1$ c_a is usually represented in a more simple way, e.g. as a point case.

Summarizing, abstraction was at all time a generalization together with a simplification of representation formalism and class membership.

4 Consequences for Similarity Measures

In this and the following three chapters we will analyze the similarity computation between a query and a case in the case base, depending on the cases' classes. Therefore, nine combinations have to be distinguished.

	pc (CB)	aigc (CB)	adgc (CB)
pc (Query)	Case 1	Case 2	Case 3
aigc (Query)	Case 4	Case 5	Case 6
adgc (Query)	Case 7	Case 8	Case 9

These nine cases can be grouped in 3 clusters, built up by the retrieval complexity: Similarity computation between non-adgcs (Case 1, Case 2, Case 4, and Case 5) that is decsribed in the following section, similarity computation for scenarios where the case base inludes adgcs (Case 3 and Case 6) that is described in 6, and similarity computation for adgc queries (Case 7, Case 8, and Case 9) that is described in 7.

For the analysis we present typical similarity measures which can of course differ in concrete applications, so they should be understood as examples, but not as the only possible ones.

4.1 Preliminaries

As we have restricted our schema to cases in attribute-value representation, we use the well-known approach to compute the case similarity by combining local similarities with an aggregation function (see [30, 31]). A case is described like introduced in section 3.1 by its attribute space $\mathbb{A}$.

Similarity Measure Approach For every attribute A_i with atomic value type T_i, a local similarity measure is defined in the following way: $sim_{Ai}(x_i, y_i) : T_i \times T_i \rightarrow [0, 1]$. For every attribute of a gc a local similarity measure of the following way is considered: $sim^*_{Ai}(x_i, y_i) : 2^{T_i} \times 2^{T_i} \rightarrow [0, 1]$.

A global similarity measure sim can than be calculated for pcs with the help of a monotonous aggregation function $F : [0, 1]^n \rightarrow [0, 1]$ like weighted average, maximum, or minimum, in a way that

$$\begin{aligned} sim(x, y) &= sim((x_1, \ldots x_n), (y_1, \ldots y_n)) \\ &= F(sim_{A1}(x_1, y_1), \ldots, sim_{An}(x_n, y_n)) \end{aligned}$$

Semantic of Sets Intervals and sets in the query as well as in the cases may have different semantics. The semantics of an interval or set in the query can be that the user specifies it because the concrete value does not matter to him or her and every value or subset within it would be acceptable (imprecise knowledge) or because he or she searches for a case, covering it (precise knowledge). The semantics of an interval or set in the case can be that either all values are supported (precise knowledge) or one or several are supported but it is not known which ones (imprecise knowledge). The following definition is inspired by Tautz and Althoff [32].

Definition 6. *If the semantics of an interval or a set is that all specified values are addressed and must be true simultaneously we speak of* precise knowledge. *If only a subset is addressed we speak of* imprecise knowledge.

5 Similarity calculation between non-adgcs

In this section, the well-understood similarity calculation between non-adgcs is summarized, thus it deals with Case 1, Case 2, Case 4 and Case 5 of the roadmap presented in 4. The most general case is Case 5, namely the calculation of similarity between an aigc query and an aigc case. The other cases are specializations, which are easier to handle.

For the analysis of Case 5 we must consider the reason why the aigcs are generalized cases: is it because some attributes of atomic value type have unknown values or because some attributes have intervals or sets as concrete values, or both. The following subsections follow this discrimination.

5.1 Atomic unknown

One or more attributes with an atomic value type of query or case have unknown values. In this case, we must distinguish three cases:

1. *Only the attribute value in the case is unknown:* Here we can make use of on an optimistic, pessimistic, or predictand strategy. In an optimistic strategy we assume that unknown values argue for similarity and express that by a local similarity of 1, $sim_{Ai}(x_i, unknown) = 1$. In a pessimistic strategy we assume that unknown values argue against similarity and express that by a local similarity of 0, that means that $sim_{Ai}(x_i, unknown) = 0$. If it is possible to calculate a kind of predictand E for the similarity, we can use the predictand strategy and define the local similarity $sim_{Ai}(x_i, unknown) = E$.
2. *Only the attribute value in the query is unknown:* Here we follow again an optimistic, pessimistic, or predictand strategy, although in the majority of cases the optimistic strategy is the most convenient one. That is because the user does not define values for attributes in the query, whose contents is not important to him or her. So the contents does not care and every values is acceptable. Optimistic strategy means that $sim_{Ai}(unknown, y_i) = 1$. Pessimistic and predictand strategy may be useful in special scenarios and are defined in an analogous way.
3. *The attribute value in case and query is unknown:* Then we can again follow an optimistic, pessimistic, or predictand strategy. Optimistic strategy means that $sim_{Ai}(unknown, unknown) = 1$; pessimistic and predictand strategy are defined in an analogous way.

5.2 Intervals

The attribute value in the query and in the case are intervals. We name the query interval $x_i = [x_{ilb}, x_{iub}]$, the case interval $y_i = [y_{ilb}, y_{iub}]$, and a possible intersection $z_i = [z_{ilb}, z_{iub}]$. We have to distinguish the four combinations of the knowledge types of definition 6.

1. *imprecise query, precise case:*
   With the imprecise query one point is requested from all the possible points of the precise case. Therefore, we just have to distinguish the following two situations:
   - $z_i \neq \emptyset$: The local similarity is one, because an arbitrary element of z_i or z_i itself can be returned.
   - $z_i = \emptyset$: The local similarity must consider the "distance" from x_{ilb} to y_{iub} and x_{iub} to y_{ilb}, so that

   $$sim^*_{Ai}(x_i, y_i) = \max\left(sim_{Ai}(x_{ilb}, y_{iub}), sim_{Ai}(x_{iub}, y_{ilb})\right)$$

   Summarizing, a local similarity measure can be defined as:

   $$sim^*_{Ai}(x_i, y_i) = \begin{cases} 1 & \text{if} \quad z_i \neq \emptyset \\ \max\begin{pmatrix} sim_{Ai}(x_{ilb}, y_{iub}), \\ sim_{Ai}(x_{iub}, y_{ilb}) \end{pmatrix} & \text{otherwise} \end{cases}$$

2. *imprecise query, imprecise case:*
   The imprecise interval in the case means that a valid value exists in this range, but it is not sure where it is. Unlike the previous case we cannot assume that valid values are in the intersection of query and case interval. Therefore, we must distinguish the three strategies: The optimistic one assumes that valid points are in the intersection, that means this is the same as the previous case. The pessimistic strategy assumes that no valid point is in the intersection which leads to the local similarity

   $$sim^*_{Ai}(x_i, y_i) = \min_{\forall x_{is} \in x_i, y_{it} \in y_i} \{sim_{Ai}(x_{is}, y_{it})\}$$

   The predictand strategy calculates the probability that a valid point is in the intersection z_i, e.g. by calculating the relation between the intersection size and the case interval size. Consequently, the similarity measure is defined as:

   $$sim^*_{Ai}(x_i, y_i) = \begin{cases} \frac{|y_{iub} - y_{ilb}|}{|z_{iub} - z_{ilb}|} & \text{if} \quad z_i \neq \emptyset \\ 0 & \text{otherwise} \end{cases}$$

3. *precise query, precise case:*
   With the precise query a case is requested whose attribute value covers the complete interval and the precise case specifies an interval with all valid points. Therefore, we have to distinguish the following four situations:
   - $x_i \subseteq y_i$: From the geometrical point of view this means that $y_{ilb} \leq x_{ilb} \leq x_{iub} \leq y_{iub}$. So y_i is a coverage for x_i, and therefore the similarity is one.
   - $x_i \supseteq y_i$: From the geometrical point of view that means that $x_{ilb} \leq y_{ilb} \leq y_{iub} \leq x_{iub}$. The similarity must be lower than one because a coverage was demanded. The similarity can be defined by the percentage of coverage: $sim^*_{Ai}(x_i, y_i) = \frac{|y_{iub} - y_{ilb}|}{|x_{iub} - x_{ilb}|}$.
   - $x_{ilb} \leq y_{ilb} \leq x_{iub} \leq y_{iub}$ or $y_{ilb} \leq x_{ilb} \leq y_{iub} \leq x_{iub}$. This case can be reduced to the preceding case by setting $y_{iub} := x_{iub}$ or $y_{ilb} := x_{ilb}$, respectively.

– $z_i = \emptyset$: From the geometrical point of view that means that $x_{iub} < y_{ilb}$ or $y_{iub} < x_{ilb}$. The similarity is zero.

Summarizing, a local similarity measure can be defined with respect to the bound as

$$sim^*_{Ai}(x_i, y_i) = \begin{cases} 1 & \text{if} \quad x_i \subseteq y_i \\ 0 & \text{if} \quad z_i = \emptyset \\ \frac{|y_{iub} - y_{ilb}|}{|x_{iub} - x_{ilb}|} & \text{otherwise} \end{cases}$$

An alternative similarity measure for software engineering experience is defined by Tautz and Althoff [32].

4. *precise query, imprecise case:*
From the logical point of view the local similarity function can never equal one. Therefore, a concrete definition is very difficult and strongly depends on the application. It can be approximated by usage of the above defined similarity measure for precise query and precise case. Of course, this is not correct, but all cases in the case base have the same problem and therefore are comparable again. Tautz and Althoff [32] defined for this situation an alternative similarity measure.

For the similarity computation in Case 2 (sim between pc query and aigc case), a way to compare a number (from the query) with an interval (from the case) must be found. This is quite simple, because a value x_{i1} can easily be regarded as an interval with coincident lower and upper bounds. The local similarity between the two intervals $x_i = [x_{ilb}, x_{iub}] = [x_{i1}, x_{i1}]$ and $y_i = [y_{ilb}, y_{iub}]$ can than be calculated like described above in the general case.

The same strategy can be applied in Case 4 (sim between aigc query and pc case). Instead of the query point the case point is regarded as an interval and the comparison of two intervals can be performed like described above.

5.3 Sets

The attribute value in the query and in the case are sets. We name the query set $x_i = \{x_{i1}, \ldots, x_{in}\}$, the case set $y_i = \{y_{i1}, \ldots, y_{im}\}$, and a possible intersection $z_i = \{z_{i1}, \ldots, z_{ik}\} = x_i \cap y_i$. We have to distinguish the four combinations of the knowledge types of definition 6.

1. *imprecise query, precise case:*
With the imprecise query one point is requested from all the possible points of the precise case. Therefore, we have to distinguish the following two situations:
   – $z_i \neq \emptyset$: Analogously to intervals, the local similarity is one.
   – $z_i = \emptyset$: The local similarity must consider the "distances" from the elements of x_i to the ones of y_i. Such a similarity function could be defined as $sim^*_{Ai}(x_i, y_i) = \max\{sim_{Ai}(x_{is}, y_{it}) \forall s \in [1, n], t \in [1, m]\}$.

   Summarizing, a local similarity measure can be defined as

$$sim^*_{Ai}(x_i, y_i) = \begin{cases} 1 & \text{if} \quad z_i \neq \emptyset \\ \max\limits_{\forall s \in [1,n], t \in [1,m]} \{sim_{Ai}(x_{is}, y_{it})\} & \text{otherwise} \end{cases}$$

2. *imprecise query, imprecise case:*
   This is analogous to intervals where the three strategies must be distinguished:

$$\text{optimistic: } sim^*_{Ai}(x_i, y_i) = \begin{cases} 1 & \text{if} \quad z_i \neq \emptyset \\ \max\limits_{\forall \binom{s \in [1,n]}{t \in [1,m]}} \{sim_{Ai}(x_{is}, y_{it})\} & \text{otherwise} \end{cases}$$

$$\text{pessimistic: } sim^*_{Ai}(x_i, y_i) = \min_{\forall x_{is} \in x_i, y_{it} \in y_i} \{sim_{Ai}(x_{is}, y_{it})\}$$

$$\text{predictand: } sim^*_{Ai}(x_i, y_i) = \begin{cases} \frac{|y_i|}{|z_i|} \text{ if} \quad z_i \neq \emptyset \\ 0 \quad \text{otherwise} \end{cases}$$

3. *precise query, precise case:*
   This situation is again analogous to the intervals; several cases can be distinguished. We are summarizing here only the resulting definition of a local similarity measure.

$$sim^*_{Ai}(x_i, y_i) = \begin{cases} 1 & \text{if} \quad x_i \subseteq y_i \\ \frac{m}{n} & \text{if} \quad x_i \supseteq y_i \\ \frac{k}{n} & \text{if} \quad z_i \neq \emptyset \text{ and } x_i \setminus y_i \neq \emptyset \text{ and } y_i \setminus x_i \neq \emptyset \\ 0 & \text{if} \quad z_i = \emptyset \end{cases}$$

   A more sophisticated similarity measure considers the "distances" between the elements. This method was introduced by Bergmann and Eisenecker [33]:

$$sim_{Ai}(x_i, y_i) = \frac{1}{n} \sum_{s=1}^{n} \max \{sim_{Ai}(x_{is}, y_{it}) \forall t \in [1, m]\}$$

   By the usage of this similarity measure the similarity for the case $z_i = \emptyset$ must not be zero.
4. *precise query, imprecise case:* With the same argument as for intervals we can again use the similarity function defined in the latter case as approximation.

For the similarity computation in Case 2 (sim between pc query and aigc case), the concrete value of the query must be regarded as a set ($x_i = \{x_{i1}\}$), then the similarity between x_i and $y_i = \{y_{i1}, \ldots, y_{im}\}$ can be calculated like described above in the general case.

The same strategy can be applied in Case 4 (sim between aigc query and pc case). Instead of the query point the case point is regarded as a set and the comparison of two sets can be performed like described above.

5.4 Non Atomic Unknown

Until now we have handled "unknown" only on attributes with atomic value type. Now we will consider the situation how the similarity between unknown and an interval or set can be computed. Therefore, we have analyzed all possible knowledge type combinations and strategies which are presented in Table 1.

In general, "unknown" for attributes with non atomic types can be handled similarly as for attributes with atomic types. The only difference occurs in the case that the query has the value unknown and the case has a set or interval. In this case the knowledge

query	case	iq & ic			iq & pc			pq & ic			pq & pc		
		O	Pe	Pr	O	Pe	Pr	O	Pe	Pr	O	Pe	Pr
unknown	interval or set	1	1	1	1	1	1	0	0	0	0	0	0
interval or set	unknown	1	0	E	1	0	E	1	0	E	1	0	E
unknown	unknown	1	0	E	1	0	E	1	0	E	1	0	E

iq imprecise query
ic imprecise case
pq precise query
pc precise case
O optimistic strategy
Pe pessimistic strategy
Pr predictand strategy

Table 1. Unknown in Sets or Intervals

types have to be distinguished, because an imprecise query always leads to a similarity of one in each strategy and a precise query always leads to zero. Depending on the knowledge types in query and case and the appropriated strategy, the corresponding sim^*_{Ai} for intervals and sets have to be adapted to take the handling of "unknown" into account.

6 Similarity calculations for case bases including adgcs

This section deals with Case 3 and Case 6 of the roadmap presented in 4, thus the similarity calculation for scenarios where the case base includes adgcs. The more general case is Case 6, namely the calculation of similarity between an aigc query and an adgc case. Case 3 (sim between pc query and adgc case) is a specialization of it.

$$\text{Case 3: } \sup_{\forall y_t \in y} \{sim(x, y_t)\} \qquad \text{Case 6: } \sup_{\forall x_s \in x, y_t \in y} \{sim(x_s, y_t)\}$$

For these cases, the similarity computation is very difficult because the presented problem must be solved. In these equations the association is used that a generalized case is a possibly infinite set of point cases and we are searching the two nearest points. For Case 3, this equation simplifies a little, because the query is a pc instead of an aigc.

Research in this area started a few years ago, but generally accepted assertions or standardized approaches cannot be given for all data types. But the following two methods are very promissing and solve the problem for special secnarios.

In 2002 Bergmann and Mougouie [34] presented a solution to calculate the similarity between a pc query and an adgc (Case 3) that is represented through constraints over an n-dimensional real-valued vector space. They have shown that the difficulty of this calculation depends on whether the adgc spawn a convex or nonconvex subspace which is defined by the constraints. For convex constraints and by usage of convex similarity measures, the Topkis-Veinott method can be easily applied to determine exactly the similarity between a pc query and an adgc. If the similarity measure is nonconvex or the adgc contains also nonconvex constraints, the problem is more difficult. For this situation an algorithm is proposed that allows to incrementally compute sequences of upper and lower bounds for the similarity and assures the convergence of the algorithm. It allows to rank the adgc in the case base according to their similarity to the pc query without the exact computation of all similarity measures.

This method is in principal also applicable in Case 6, except that it only works for $unknown$ values; intervals and sets are not possible for the query at the current state.

Currently, we investigate [35] another more pragmatic strategy to handle this problem: a converter that samples the subspace spawned by the adgc and creates a number of pcs that are distributed "reasonably" within the subspace. After the conversion of every adgc, the new case base consists obviously of much more cases, but only of pcs. Consequently, standard algorithms for the similarity computation between two point cases can be used. This method is applicable in Case 3 and Case 6.

7 Similarity calculations for adgc queries

This section deals with Case 7, Case 8, and Case 9 of the roadmap presented in 4, thus the similarity calculation for scenarios with adgc queries. These cases have not been a topic of research until now. One big problem which has to be solved is to find a possibility to define attribute dependent queries in a user friendly way. Assumed, such a query could be defined – albeit only for simple dependencies – we face the problem to compute the similarity of this query to the cases. This problem may be solved by reducing it to well-known similarity computations, e.g. by changing the cases' model or by converting the query to several point queries, search the most similar cases to all of them, and return a unified set of cases.

Example: Imagine a car-selling domain with an attribute "features", whose type is a set of features like "electronic exterior mirrors", "winter equipment", "rear wing", or "large tires". The user may specify implications like "If the car has electronic exterior mirrors, it should also have winter equipment" and "If the car has a rear wing, it should also have large tires". One solution is to change the representation of the feature attribute from the set to a multiset. The implications can than be converted in the set $\{\{$electronic exterior mirrors, winter equipment$\}, \{$rear wing, large tires$\}\}$, so that we have an aigc query. Another workaround is to request two aigc queries: one with the feature attribute $\{$electronic exterior mirrors, winter equipment$\}$ and the second one with $\{$rear wing, large tires$\}$.

8 Summary and Outlook

We have presented an integrated classification schema for cases and have described the similarity computation, depending on the class membership of query and case. We have elaborated that the similarity computation with adgcs is not well researched until now. Hence, the current research should focus on this topic.

The similarity computation may be solved by reducing it to well known similarity computations as touched on in 7. Another idea could be to abstract adgcs with the goal that it becomes an aigc or pc on a higher level of abstraction.

A completely different approach would be the definition of similarity measures on constraints which would enable the comparison of the adgcs' shapes.

This approach is motivated by the different knowledge types (definition 6): imprecise adgc queries can be handled similar to 5, but precise adgc queries can be subdivided in a "coverage request" and a *shape request*. The request of a similar shape leads to the new question how the similarity between constraints, which define the shapes, can be computed. This is not only a theoretical playing, because applications for such scenarios

are possible by all means. For example, a designer of furniture searches for a technique to realize a parameterized furniture. In this case the parameterization is the query and the results should be furniture with the same parameterization.

References

1. Schank, R.C., Abelson, R.P.: Scripts, plans, goals and understanding: An inquiry into human knowledge structures. Lawrence Erlbaum Associates, Hillsdale, NJ (1977)
2. Kolodner, J.L.: Retrieval and Organizational Strategies in Conceptual Memory. PhD thesis, Yale University (1980)
3. Kolodner, J.L.: Case-Based Reasoning. Morgan Kaufmann, San Mateo (1993)
4. Bareiss, R.: Exemplar-Based Knowledge Acquisition: A unified Approach to Concept Representation, Classification and Learning. Academic Press (1989)
5. Salzberg, S.: A nearest hyperrectangle learning method. Machine Learning **6** (1991) 277–309
6. Schank, R.C.: Dynamic Memory: A Theory of Learning in Computers and People. Cambridge University Press, New York (1982)
7. Kolodner, J.L., Simpson, R.L.: The MEDIATOR: Analysis of an Early Case-Based Problem Solver. Cognitive Science **13** (1989) 507–549
8. Hua, K., Smith, I., Faltings, B.: Integrated case-based building design. In Wess, S., Althoff, K.D., Richter, M.M., eds.: Topics in Case-Based Reasoning. Proc. Of the First European Workshop on Case-Based Reasoning (EWCBR-93). Lecture Notes in Artificial Intelligence, 837, Springer Verlag (1993) 436–445
9. Gebhardt, F., Voß, A., Gräther, W., Schmidt-Belz, B.: Reasoning with Complex Cases. Volume 393 of International Series in Engineering and Computer Science. Kluwer Academic Publishers, Boston (1997)
10. Bakhtari, S., Bartsch-Spörl, B., Oertel, W., Eltz, U.: DOM: domain ontology modelling in architectural and engineering design. Fabel-Report 33, GMD, Sankt Augustin (1995)
11. Oertel, W., Bakhtari, S.: Interaction of generic knowledge and cases in DOM. In: Proceedings of the Third Congress on Computing in Civil Engineering, Anaheim, USA (1996)
12. Bergmann, R.: Effizientes Problemlösen durch flexible Wiederverwendung von Fällen auf verschiedenen Abstraktionsebenen. DISKI 138. infix (1996)
13. Bergmann, R., Wilke, W.: On the role of abstraction in case-based reasoning. In Smith, I., Faltings, B., eds.: Advances in Case-Based Reasoning. Lecture Notes in Artificial Intelligence, 1186, Springer Verlag (1996) 28–43
14. Rayner, N., Harris, C.: Mission management for multiple autonomous vehicles (1996)
15. Aamodt, A., Plaza, E.: Case-based reasoning: foundational issues, methodological variations, and system approaches. AI Communications **7** (1994) 39–59
16. Reinartz, T., Iglezakis, I., Roth-Berghofer, T.: On quality measures for case base maintenance. In Blanzieri, E., Portinale, L., eds.: Advances in Case-Based Reasoning (EWCBR'2000). Lecture Notes in Artificial Intelligence, 1898, Springer (2000) 247ff
17. Reinartz, T., Iglezakis, I., Roth-Berghofer, T.: On quality measures for case-base maintenance. Computational Intelligence – Special Issue on Maintaining CBR Systems **17** (2001)
18. Schmidt, R., Gierl, L.: The Roles of Prototypes in Medical Case-Based Reasoning Systems. In Burkhard, H.D., Lenz, M., eds.: 4th German Workshop on CBR — System Development and Evaluation —. Informatik-Berichte, Berlin, Humboldt University (1996) 207–216
19. Schmidt, R., Gierl, L.: Case-based reasoning for antibiotics therapy advice: An investigation of retrieval algortihms and prototypes. Artificial Intelligence in Medicine **23** (2001) 171–186

20. Lewis, J.: Intellectual property (IP) components. Artisan Components, Inc., [web page], http://www.artisan.com/ip.html (1997) [Accessed 28 Oct 1998].
21. Bergmann, R., Vollrath, I., Wahlmann, T.: Generalized cases and their application to electronic designs. In Melis, E., ed.: 7. German Workshop on Case-Based Reasoning (GWCBR'99). (1999) 6–19
22. Smyth, B., Cunningham, P.: Déjà Vu: a hierarchical case-based reasoning system for software design. In Neumann, B., ed.: ECAI 92: 10th European Conference on Artificial Intelligence, August 1992, Vienna. Wiley, Chichester (1992) 587–589
23. Smyth, B.: Case-Based Resign. PhD thesis, Trinity College Dublin (1996)
24. Kambhampati, S., Hendler, J.: A validation-structure-based theory of plan modification and reuse. Artificial Intelligence **55** (1992) 193–258
25. Bergmann, R., Wilke, W.: Building and refining abstract planning cases by change of representation language. Journal of Artificial Intelligence Research **3** (1995) 53–118
26. Bergmann, R.: On the use of taxonomies for representing case features and local similarity measures. In Gierl, L., Lenz, M., eds.: Proceedings of the 6th German Workshop on Case-Based Reasoning. (1998)
27. Branting, K.L., Aha, D.W.: Stratified case based reasoning: Reusing hierarchical; problem solving eposides. Technical Report AIC-95-001, Naval Research Lab (1995)
28. Karchenasse, N., Roger, J.M., Sevila, F.: The hierarchical case-based diagnosis. In Bergmann, R., Wilke, W., eds.: Proceedings of the 5th German Workshop ion Case-Based Reasoning, LSA 97-01E, University of Kaiserslautern, Germany (1997)
29. Cunningham, P., Bonzano, A.: Hierarchical CBR for multiple aircraft conflict resolution. In Prade, H., ed.: Proceedings of the 13th European Conference on Artificial Intelligence, Wiley (1998)
30. Bergmann, R.: Experience Management - Foundations, Development Methodology, and Internet-Based Applications. Lecture Notes in Artificial Intelligence 2432. Springer Berlin, Heidelberg, New York, Hong Kong, London, Milan, paris, Tokyo (2002)
31. Wess, S.: Fallbasiertes Problemlösen in wissensbasierten Systemen zur Entscheidungsunterstützung und Diagnostik. PhD thesis, Universität Kaiserslautern (1995) Available as DISKI 126, infix Verlag.
32. Tautz, C., Althoff, K.D.: Towards engineering similarity functions for software engineering experience. In Göker, M., ed.: 8. German Workshop on Case-Based Reasoning (GWCBR'2000)., DaimerChrysler Research Ulm (2000) 87–97
33. Bergmann, R., Eisenecker, U.: Fallbasiertes schließen zur unterstützung der wiederverwendung objektorientierter software: Eine fallstudie. Technical report, University of Kaiserslautern, Kaiserslautern (1994)
34. Mougouie, B., Bergmann, R.: Similarity assessment for generalizied cases by optimization methods. In: Proceedings of the European Conference on Case-Based Reasoning (ECCBR-02), Springer. (2002)
35. Maximini, R., Tartakovski, A., Bergmann, R.: Investigating different methods for efficient retrieval of generalized cases. In: German Workshop on Experience Management. (2003) accepted.