The Conflict Graph for Maintaining Case–Based Reasoning Systems

Ioannis Iglezakis

DaimlerChrysler AG, Research & Technology, FT3/AD,
P.O. Box 2360, 89013 Ulm, Germany
ioannis.iglezakis@daimlerchrysler.com

Abstract. The maintenance of case–based reasoning systems is remarkably important for the continuous working ability of case–based reasoning applications. To ensure the utility of these applications case properties like correctness, consistency, incoherence, minimality, and uniqueness are applied to measure the quality of the underlying case base. Based on the case properties, the conflict graph presents a novel visualization of conflicts between cases and provides a technique on how to eliminate these conflicts and therefore maintain the quality of a case base. An evaluation on ten real world case bases shows the applicability of the introduced technique.

1 Introduction

During the last years, the research in the realm of case–based reasoning (CBR) systems in general and maintaining of case–based reasoning systems in particular has become a main topic for many researchers. In the field of maintaining case–based reasoning systems various measures like coverage and reachability [14,17] and thereby case competence [15] are modeling the competence of a case base. The use of case properties which show the relations within cases and between cases [11] is an other way to measure the quality of a case base. Compared to the standard case–based reasoning cycle [1], the case properties are embedded in an extended case–based reasoning cycle [12] with the two additional steps: *review* and *restore*. The review step detects conflicts within and between cases with the help of the case properties. These conflicts are then monitored and the decision is made when to maintain and therefore trigger the restore step. The restore step suggests the necessary changes for the contents of the case base to reestablish the quality.

This paper focuses on the monitor task of the review step and how it can be used to restore the quality of the case base. It introduces a new visualization for the conflicts between cases — the conflict graph — and shows how this conflict graph can be used to find a subset of cases which must be changed in order to restore the quality of a case base. Therefore the needed case properties are revisited in section 2. The concept of a conflict graph is defined in section 3, followed by results of an evaluation in section 4. Section 5 discusses the related work. Finally, section 6 concludes the paper with a summary and issues of further work.

2 The Case Properties Revisited

As explained above, the review step in the extended case–based reasoning cycle has among other things like monitoring the quality of the case base the duty of measuring the case base through case properties. These measures are disjunct and used for quality indication within cases or between cases. The formal definition of the case properties which are used here can be found in [11]. In this section only an informal description of the case properties *correctness*, *consistency*, *uniqueness*, *minimality*, and *incoherence* is given to motivate them for further use in the next sections.

2.1 Correctness

There are two kinds of case properties: *isolated* and *comparative*. Correctness is the only isolated property, because the correctness of a case can be recognized without considering any other case. A case is believed to be correct if and only if the problem description of the case corresponds to the given solution. This implies that the solution solves the problem that is described by the problem description. Note that for the following descriptions of case properties and experiment the correctness of the cases is assumed.

2.2 Consistency

The first of the comparative case properties is consistency. A given case is called consistent if and only if there is no other case whose problem description is a subset or the same of the given case's problem description and their solutions are different. Hence, consistency covers situations where the subset relation between problem description with different solutions reports that there possibly is an absence of some attributes or values to make the two cases distinctive.

Table 1 illustrates a counter–example from the help–desk domain when two cases are not consistent. This example shows two cases with the same operating system, the

Table 1. Example for (in-) consistency

Operating System	Printer	Printing	Paper	Ink	Solution
WinNT	HP840c	No	Yes	Yes	*Update Driver*
WinNT	HP840c	No	—	—	*Second Level*

same printer and no printing. In addition, the first case has more information about the paper and ink situation than the second. Note that the value "—" for the paper and ink attribute in the second case means the these values are unknown. Thus, with regard to the above description the second case is not consistent, because the problem description of the second case is a real subset of the first case and the solutions are different.

In general, consistency contains the occurrence of alternatives. It is assumed that for each capable problem description there exists a “best” solution. As a matter of course, the case base should only consist of cases that have “best” solutions. Thus, no alternative solutions are allowed neither as an alternative in the same case nor as a further case with the same or a more general problem description and a different solution.

Note that it is possible for some domains to have equivalent solutions for the same or more general problem description. The travel domain is a good example for this kind of domain and it is up to the case base administrator to decide whether these alternatives are admitted or not.

2.3 Uniqueness

Another comparative case property is uniqueness. A given case is called unique if and only if the problem description and corresponding solution of each other case in the case base is different. A counter–example from the help–desk domain is displayed in table 2. It describes the fact when the given case property uniqueness is violated.

Table 2. Example for (not) unique

Operating System	Printer	Printing	Paper	Ink	Solution
WinNT	HP840c	No	Yes	Yes	Update Driver
WinNT	HP840c	No	Yes	Yes	Update Driver

For both cases in this example, the problem descriptions and solutions are exactly the same. With the above description of uniqueness follows that both cases are evidently not unique.

2.4 Minimality

The third comparative case property is minimality which means the opposite of subsumption. A given case is called minimal if and only if there is no other case for which the problem description is a real subset of the given case and the solutions are identical. This concept is clarified by the counter–example in table 3 which contains an example when two cases are not minimal.

In this table both cases are identical besides the value for the operating system. The first case contains a value for the operating system while the second case does not. This disobeys the definition of minimality because the problem description of the second case is a real subset of the first case’s problem description and the solution for both cases are equal.

Table 3. Example for (not) minimal

Operating System	Printer	Printing	Paper	Ink	Solution
WinNT	HP840c	No	Yes	Yes	Update Driver
—	HP840c	No	Yes	Yes	Update Driver

2.5 Incoherence

Finally, the last comparative case property is incoherence. A given case is incoherent if and only if each other case with the same solution as the given case has the same problem description with the exception of a specific (small) number (Δ) of attributes whose values are different. To illustrate this case property in table 4 once more a counter–example is used.

Table 4. Example for (not) incoherent (for $\Delta = 1$)

Operating System	Printer	Printing	Paper	Ink	Solution
WinNT	HP840c	No	Yes	Yes	Update Driver
Win95	HP840c	No	Yes	Yes	Update Driver

Both cases in table 4 have the same solution and differ only in the value for the operating system and in contrast to the minimality example in both cases these values exist. With the assumption that $\Delta = 1$ and the above description it follows that these two cases have a conflict with the definition of incoherent and are therefore not incoherent.

This section gave an informal description of the case properties in combination with examples from the help–desk domain. With these descriptions it is possible to build case base properties in which the case properties are assigned to a whole case base. For example, if each case in a case base is minimal, then the entire case base is by definition also minimal. To receive a better granularity of assessment, the case properties can be measured in degrees within the case base. The measurement for each case property is computed by counting the number of cases which fulfill this property divided by the total number of cases in the case base. This mapping of sets to numbers offers a more sophisticated way to measure the quality of the case base. Finally, quality measures are defined as a function over the degrees of case properties. For example, a weighted sum could respresent one usable function. More information on the subject of quality measures for case base maintenance can be found in [11]. In the following section, the notion of the conflict graph is introduced. This notion can help to visualize the conflicts between cases and decide which cases should be changed in order to restore the quality of a case base.

3 The Conflict Graph

After measuring the quality of the case base with the help of the case properties, the next step is to monitor these results. This section will introduce a novel method to display conflicts between cases which are indicated by case properties and how this representation can be used in order to maintain a case base.

When applying the described case properties from the previous section to the case base it is usually that not all properties are satisfied and thus the degree of case properties is below 100%. This means that for some cases a case property is not fulfilled. The purpose of the *single conflict indicators* in definition 1 is to test two cases for a particular case property and to indicate a possible conflict.

Definition 1 (Single Conflict Indicator). *Assume a case base C with two cases $c_i, c_j \in C$ and $i \neq j$.*

$$
\begin{aligned}
&(i) \quad cInd_1 : C \times C \mapsto \{\neg\, consistent, \emptyset\},\\
&\qquad cInd_1(c_i, c_j) := \begin{cases} \neg\, consistent, \ if\ c_i\ is\ not\ consistent\ with\ c_j \\ \emptyset\ else \end{cases}\\
&(ii) \quad cInd_2 : C \times C \mapsto \{\neg\, incoherent, \emptyset\},\\
&\qquad cInd_2(c_i, c_j) := \begin{cases} \neg\, incoherent, \ if\ c_i\ is\ not\ incoherent\ with\ c_j \\ \emptyset\ else \end{cases}\\
&(iii)\ cInd_3 : C \times C \mapsto \{\neg\, minimal, \emptyset\},\\
&\qquad cInd_3(c_i, c_j) := \begin{cases} \neg\, minimal, \ if\ c_i\ is\ not\ minimal\ with\ c_j \\ \emptyset\ else \end{cases}\\
&(iv) \quad cInd_4 : C \times C \mapsto \{\neg\, unique, \emptyset\},\\
&\qquad cInd_4(c_i, c_j) := \begin{cases} \neg\, unique, \ if\ c_i\ is\ not\ unique\ with\ c_j \\ \emptyset\ else \end{cases}
\end{aligned}
$$

To explain the above as well as the following definitions an exemplary case base is designed. Table 5 shows this case base with seven cases. Each case c_i contains the solution s_i and the problem description p_i which encloses the attribute–value pairs v_{ij}.

Table 5. Example case base with seven cases

c_i		p_i		s_i
c_1	v_{11}			s_1
c_2	v_{12}	v_{22}		s_1
c_3	v_{11}	v_{22}		s_1
c_4	v_{12}	v_{22}	v_{33}	s_2
c_5	v_{11}	v_{23}	v_{35}	s_1
c_6	v_{11}	v_{23}	v_{34}	s_1
c_7	v_{11}			s_1

Example 1 (Single Conflict Indicator). Assume the cases c_1 and c_6 from the case base given in table 5 and the single conflict indicators $cInd_1$ and $cInd_3$. The results are: $cInd_1(c_1, c_6) = \emptyset$ and $cInd_3(c_1, c_6) = \neg\, minimal$.

After defining the single conflict indicators the next step is to define the *conflict indicator set* which is then used to build the *conflict indicator*.

Definition 2 (Conflict Indicator Set). *Assume the indices of all single conflict indicators $I = \{1, \ldots, n\}$, where n is the maximum number of single conflict indicators, and $I^{\subseteq}$ the set of all subsets of I. The conflict indicator set M is defined as $M \in I^{\subseteq}$ with $M \neq \emptyset$ and $m_l \in M$ where l denotes the l–th element of M.*

The purpose of the conflict indicator set is to choose which single conflict indicators are used to configure a conflict indicator. Note that for the single conflict indicators given in definition 1 the maximum number n of single conflict indicators is $n = 4$ which limits the possible number of different conflict indicator sets to $2^n - 1 = 15$.

Definition 3 (Conflict Indicator). *Assume a case base C with two cases $c_i, c_j \in C$ and $i \neq j$, and M is a conflict indicator set.*

$$cInd : I^{\subseteq} \times C \times C \mapsto \{\neg\, consistent, \neg\, incoherent, \neg\, minimal, \neg\, unique, \emptyset\},$$
$$cInd(M, c_i, c_j) := \bigcup_{l=1}^{|M|} cInd_{m_l}(c_i, c_j)$$

Note that for the given single conflict indicators $cInd_k$ in definition 1 the value of $|cInd(M, c_i, c_j)|$ is 0 or 1 for all conflict indicator sets M and pairs of cases c_i, c_j.

The conflict indicator is a composition of the single conflict indicators. The conflict indicator set M determines the configuration of which single conflict indicators are used. The results are the kind of conflicts between two cases or the empty set if all case properties are satisfied. Example 2 illustrates this behavior.

Example 2 (Conflict Indicator). Assume the cases c_1 and c_6 from the case base given in table 5 and the conflict indicator set $M = \{1, 2\}$ and $M' = \{3, 4\}$. Then results are $cInd(M, c_1, c_6) = \emptyset$, and $cInd(M', c_1, c_6) = \neg\, minimal$.

The *conflict graph* creates a graphical representation of conflicts between two cases within a case base which are indicated through the conflict indicator. This means that the conflict graph contains all the cases which have conflicts detected by the case properties and connects these cases with a labeled edge. The label depends on the kind of property violation. Definition 4 formalizes this in a more detailed form.

Definition 4 (Conflict Graph). *Assume a case base $C = \{c_1, \ldots, c_i\}$, a conflict indicator set M, a set of nodes $N \in C^{\subseteq}$ where $C^{\subseteq}$ is the set of all subsets of C, and a set of edges $E \subseteq N \times N$ with $e_{jk} \in E$ and e_{jk} defines an edge from node of case c_j to node of case c_k with label $cInd(M, c_j, c_k)$ if and only if $cInd(M, c_j, c_k) \neq \emptyset$. The conflict graph $G = (N, E, M)$ is a directed Graph with labeled edges and $\forall c_j \in N, \exists c_k \in N : cInd(M, c_j, c_k) \neq \emptyset$ and $\forall c_j \in C \setminus N, \forall c_k \in C : cInd(M, c_j, c_k) = \emptyset$.*

Note that for some case bases it is possible to have more than one conflict graph which are not connected. For example, if a case base has four cases $\{c_1, \ldots, c_4\}$ and c_1 has only an inconsistency conflict with case c_2, case c_3 has only a minimality conflict with case c_4 and no other conflicts are detected, then the result would be two conflict graphs with the cases c_1 and c_2 in the one and the cases c_3 and c_4 in the other graph.

Figure 1 shows the conflict graph for the example case base given in table 5. The graph displays thirteen conflicts within the case base. One consistency conflict (c_2, c_4), four incoherence conflicts $(c_2, c_3), (c_3, c_2), (c_5, c_6), (c_6, c_5)$, six minimality conflicts $(c_1, c_3), (c_1, c_5), (c_1, c_6), (c_7, c_3), (c_7, c_5), (c_7, c_6)$, and two uniqueness conflicts which are (c_1, c_7) and (c_7, c_1). Note that with the increasing number of cases the conflict graph

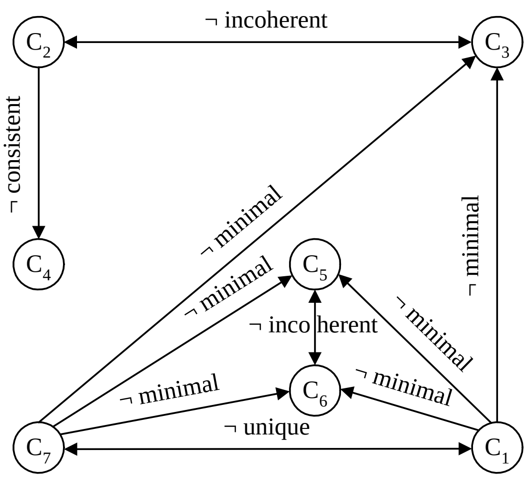

Fig. 1. The conflict graph for table 5

can confuse a human observer like the case base administrator. However it is possible to cluster groups of conflict cases and present these groups first with the ability to navigate into these groups.

The advantage of the conflict graph in comparison to other representations like adjacent lists is the ability to display the cases which do not satisfy the defined case properties in a compact manner. Thus, the case base administrator can decide which cases have to be modified to restore the quality of the case base. However this decision is very complex because the goal is to modify at least as possible cases dependable on the arising costs. The following definitions do support the case base administrator within this task by identifying the cases which should be modified. Hence, the following definitions can help to automate the parts of the restore step within the extended case base reasoning cycle.

Definition 5 (Independent Graph). *A Graph $G = (N, E, M)$ with conflict indicator set M is independent if and only if there is no pair of nodes $n_i, n_j \in N$ for which G defines an Edge $e_{ij} \in E$.*

Graph $G = (N, E, M)$ is independent $:\iff E = \emptyset$

An *independent graph* is the result after the decision is made which cases have to be modified. It is a graph in which the nodes have no edges and therefore no conflicts which each other. For example, if the cases $\{c_1, c_3, c_4, c_5, c_6\}$ are removed from the conflict graph in figure 1 then the resulting graph would be independent. However it is not appropriate to remove too many cases because of the unnecessary loss of knowledge and the costs which appear with each modification of the case base. Note that instead of removing a case other modify operators are possible. For simplicity, in this paper only the remove operator is considered. A list of more sophisticated modify operators can be found in [12].

Definition 6 (Optimal Subset). *Assume the conflict graph $G = (N, E, M)$, the set $N^{\subseteq}$ of all subsets of N, the conflict indicator set M, the cost function $cost : N^{\subseteq} \mapsto [0; 1]$, $N' \in N^{\subseteq}$, and the graph $G' = (N', E', M)$ is independent.*

N' is the optimal subset within $N^{\subseteq}$ $:\iff$ $\nexists N'' \in N^{\subseteq} : cost(N') > cost(N'')$ and the graph $G'' = (N'', E'', M)$ is independent.

Definition 6 states that the optimal subset of a conflict graph creates an independent graph and is mainly dependable of a cost function. The cost function is an arbitrary function which describes the optimization goal. Example 3 introduces a cost function for which the search for an optimal subset becomes equivalent to the search for a maximum independent set.

Example 3 (Maximum Independent Set). Assume a conflict graph $G = (N, E, M)$, a node subset $N' \in N^{\subseteq}$, and a cost function $cost : N^{\subseteq} \mapsto [0; 1], cost(N') := |N'|/|N|$. Then the search for an optimal subset is equivalent to the search for a maximum independent set. For the conflict graph in figure 1 the maximum independent set is (c_3, c_4, c_5), or (c_3, c_4, c_6). Note that the search for a maximum independent set is $\mathcal{NP} - complete$ [13].

The case base administrator can control the search for an optimal subset on three different ways. If there exists more than one solution for the optimal subset like in example 3 then the first way is to decide which of the possible solution is adequate for the actual situation. The second way is to mark the cases which should be put in the optimal subset. This can totally change the behavior of the algorithm which searches for the optimal subset. For example, if case c_1 is marked for the conflict graph in figure 1, then the optimal subset with the same cost function as in example 3 is (c_1, c_2) or (c_1, c_4) and hence different from the optimal subset without any interaction. Finally, the third way is to mark the cases which should not be included in the optimal subset. Again, this would change the manner of how to search for the optimal subset. For example, if case c_5 is marked for the conflict graph in figure 1, then the optimal subset with cost function like in example 3 is (c_3, c_4, c_5).

4 Evaluation

The last section introduced the conflict graph and how it can be used to indicate a subset of cases which should be changed to restore the quality of the case base. This section will present an evaluation of these conflict graph related techniques for the maintenance of case–based reasoning systems.

4.1 Experimental Procedure

To perform the experiments, the data set of cases is divided into a training set and a test set. The training set becomes the case base and is then used for the case–based reasoning algorithm. Then each case from the test set is tested against the case base within a case–based reasoning algorithm. This implements the benchmark for the other experiments and a certain accuracy and case base size is acquired. To achieve the remaining experiments the training case base is optimized. Thus, the conflict indicator is configured with the following conflict indicator sets $\{\{1\}, \{2\}, \{3\}, \{4\}, \{1, 2, 3, 4\}\}$. This means that each single case property was applied on its own as a conflict indicator and then all properties together. For the optimization the same cost function is chosen as in example 3: $cost : N^{\subseteq} \mapsto [0; 1], cost(N') := |N'|/|N|$ where N denotes the Nodes in the conflict graph and $N' \in N^{\subseteq}$. This means that the search for the optimal subset is $\mathcal{NP} - complete$. Algorithm 1 is a heuristics to calculate a not necessarily optimal solution of the optimal subset for a given graph. The algorithm returns a nodes–collection which is then viewed as the optimal subset and the graph which can build from the nodes–collection is guaranteed to be independent.

Algorithm 1 (Heuristic to calculate the optimal subset for a given graph).

```
nodes–collection := nodes of graph
while (nodes–collection does not build an independent graph)
   n := node with the maximum number of incoming and outgoing edges
   remove node n from nodes–collection
end while
return nodes–collection
```

Furthermore, the optimization's modify operator is remove and the optimization is done automatically without human interaction. After optimizing the case base the test cases are applied to the case base, a level of accuracy is reached, and the case base size is registered.

4.2 Experimental Results

The aim of the predefined benchmarks and experiments is to show that the representation of the conflict graph and the following search for the optimal subset is applicable for maintaining case–based reasoning systems.

To accomplish this, data of the UCI Machine Learning Repository [6] is taken. Namely, the following case bases are used: *Annealing* (797 cases), *Audiology* (200 cases), *Australian* (690 cases), *Credit Screening* (690 cases), *Housing* (506 cases), *Letter Recognition* (16000 cases), *Pima* (768 cases), *Soybean* (307 cases), *Voting Records* (435 cases), and *Zoo* (101 cases). To achieve the results that are presented below, a five–fold cross validation with a basic optimistic nearest neighbor algorithm is used. If numerical attributes did occur they have been pre–processed by a standard equal–width discretization.

Table 6. Results

	benchmark		consistency		incoherence		minimality		uniqueness		all properties	
case base	size	acc.	size	acc.	size	acc.	size	acc.	size	acc.	size	acc.
Annealing	637.6	0.97	622.8	0.97	416.0	0.97	532.6	0.97	574.8	0.97	335.6	0.97
Audiology	160.0	0.70	160.0	0.70	144.2	0.70	156.0	0.70	145.2	0.69	128.6	0.70
Australian	552.0	0.81	550.6	0.81	509.4	0.80	552.0	0.81	548.6	0.81	506.0	0.80
Credit S.	552.0	0.80	550.6	0.80	508.8	0.79	551.4	0.80	549.0	0.79	505.6	0.78
Housing	404.8	0.31	404.2	0.31	391.2	0.30	404.8	0.31	401.4	0.30	387.2	0.30
Letter R.	16000.0	0.88	16000.0	0.88	14787.8	0.88	16000.0	0.88	15081.4	0.88	14044.6	0.88
Pima	614.4	0.65	614.4	0.65	575.6	0.65	614.4	0.65	612.0	0.65	573.0	0.65
Soybean	245.6	0.92	244.8	0.92	217.0	0.92	245.0	0.92	243.2	0.92	213.2	0.92
Voting R.	348.0	0.93	344.4	0.92	257.6	0.93	271.6	0.93	280.6	0.93	191.8	0.93
Zoo	80.8	0.95	80.8	0.95	53.8	0.95	80.8	0.95	50.4	0.95	33.4	0.95

Table 6 shows that optimization through the case properties keeps the case base consistent, incoherent, minimal, and unique and reduces the case base size while preserving the prediction accuracy.

For the single case properties as conflict indicators, the case base size is reduced up to 34.8% from the original case base for incoherence in the annealing case base and the reduction for all properties as conflict indicator is up to 58.7% while the prediction accuracy remains the same as in the benchmark. The combination of all case properties gives each time a better case reduction result than the single case properties. The results vary from 0.5% between incoherence and all properties in the pima case base and 33.7% between uniqueness and all properties in the zoo case base.

In addition, the letter recognition case base with 16000 cases shows that the described technique is applicable for large case bases, too.

Furthermore, in comparison with the results in [8] which uses a different kind of optimization the results for the prediction accuracy were better and thus more robust against the used case base domain. Moreover, this evaluation used combinations of the case properties while the earlier did not. Another advantage of the optimization used in this evaluation is that it is possible to have an interaction with the case base administrator.

This shows that with the help of the conflict graph and the search of an optimal subset within this graph it is possible to maintain a case base reasoning system. Furthermore with the interactive help of a case base administrator the results should improve.

5 Related Work

The formal definitions of the case properties used for the conflict graph can be found in Reinartz et al. [11]. Several additional measures like case base properties, degrees of case properties, and quality measures are defined there, too. Evaluations showing that case properties are valueable for maintaining case–based reasoning systems have been done by Iglezakis et al. [7,8]. The extension of the standard case–based reasoning cycle and the definitions of the modify operators were first proposed by Reinartz et al. [12].

The use of graphs to maintain conversational case libraries (CCL) has been presented by Aha [3] and Aha and Breslow [4,5]. The CCL is applied to a transformation process which produces hierarchies. Then the hierarchies are pruned with the help of some design guidelines provided by case–based reasoning vendors. The resulting hierarchies are transformed back into cases of the CCL.

Over the years there where different kinds of measures for the maintenance of case–based reasoning systems published. Racine and Yang [9,10] proposed the measures for inconsistency and redundancy for the maintenance of large and unstructured case bases. Furthermore, with the access and use of background knowledge it is possible to differentiate between intra–case and inter–case measures. Smyth and Keane [14] described two different measures — coverage and reachability. The coverage of a case is the set of all problems in the problem area which can be solved by this case through adaptation. The reachability of a case is the set of all cases which are used to solve this case through adaptation. These two measures lead to different strategies how to preserve the competence of the used case base, namely case deletion by Smyth and Keane, and case addition introduced by Zhu and Yang [17]. In addition, Smyth and McKenna [15] presented the measures for case density and group density for modeling competent case–based reasoning systems which are case addition strategies, too. More strategies on how to decide which cases should be stored are proposed by the DROP1–DROP5 and DEL Algorithms by Wilson and Martinez [16], and Aha's CBL1–CBL4 algorithms [2].

6 Conclusions

With the help of case properties it is possible to create consistent, incoherent, minimal, and unique case bases. These properties have been shown to be valuable in the field of maintaining case–based reasoning systems. The application of these properties is the first step in a framework for maintaining case–based reasoning systems, to monitor the properties and then if necessary to restore the quality of the case base are the next steps.

This paper introduced a novel method to visualize conflicts between cases which are detected by the case properties and how to use this presentation to maintain a case base. This is accomplished by formal definitions of the conflict graph and the optimal subset. The subsequent evaluations showed that the proposed concepts and methods are useful for maintaining case–based reasoning systems, by reducing the size of the evaluated case bases and preserve the prediction accuracy while satisfy the case properties.

Further tasks appear on both sides of the review–restore chain of the extended case–based reasoning cycle. At the beginning of the chain, the case properties can be extended by the use of the similarity measures. This would allow the application of the case properties more flexibility in use. On the other side of the chain the use of other modify operators than remove (cf. Reinartz et al.[12]) provides a strong technique for not only reducing the case base size but also for changing the cases which have conflicts and therefore increasing the quality of the case base. This would yield to an approach which does improve the cases rather than remove them.

Acknowledgments

The author is grateful to Thomas Reinartz (DaimlerChrysler, Research & Technology, FT3/AD) and Thomas Roth–Berghofer (tec:inno — empolis knowledge management GmbH) for the fruitful discussions and continual support.

References

1. Agnar Aamodt and Enric Plaza. Case–based reasoning: Foundational issues, methodological variations, and system approaches. *AI Communications*, 7(1):39–59, 1994.
2. David W. Aha. Case–based learning algorithms. In *Proceedings of the DARPA Case–Based Reasoning Workshop*, pages 147–158. Morgan Kaufmann, 1991.
3. David W. Aha. A proposal for refining case libraries. In *Proceedings of the 5th German Workshop on Case–Based Reasoning (GWCBR)*, 1997.
4. David W. Aha and Leonard A. Breslow. Learning to refine case libraries: Initial results. Technical Report AIC–97–003, Navy Center for Applied Research in AI, 1997.
5. David W. Aha and Leonard A. Breslow. Refining conversational case libraries. In *Proceedings of the Second International Conference on Case–Based Reasoning*, pages 267–278, 1997.
6. Catherine L. Blake and Christopher J. Merz. UCI repository of machine learning databases, 1998.
7. Ioannis Iglezakis and Christina E. Anderson. Towards the use of case properties for maintaining case based reasoning systems. In *Proceedings of the Pacific Knowledge Acquisition Workshop (PKAW)*, pages 135–146, 2000.
8. Ioannis Iglezakis, Thomas Roth–Berghofer, and Christina E. Anderson. The application of case properties in maintaining case–based reasoning systems. In *Professionelles Wissensmanagement: Erfahrungen und Visionen (includes the Proceedings of the 9th German Workshop on Case–Based Reasoning (GWCBR)),* pages 209–219. Shaker Verlag, 2001.
9. Kirsti Racine and Qiang Yang. On the consistency management of large case bases: the case for validation. In *Proceedings of the AAAI–96 Workshop on Knowledge Base Validation, American Association for Artificial Intelligence (AAAI)*, pages 84–90, 1996.
10. Kirsti Racine and Qiang Yang. Maintaining unstructured case bases. In *Proceedings of the 2nd International Conference on Case–Based Reasoning (ICCBR),* pages 553–564. Springer–Verlag, 1997.
11. Thomas Reinartz, Ioannis Iglezakis, and Thomas Roth–Berghofer. On quality measures for case base maintenance. In *Proceedings of the 5th European Workshop on Case–Based Reasoning*, pages 247–259. Springer–Verlag, 2000.

12. Thomas Reinartz, Ioannis Iglezakis, and Thomas Roth–Berghofer. Review and restore for case–base maintenance. *Computational Intelligence: special issue on maintaining CBR systems*, 17(2):214–234, 2001.
13. Steven S. Skiena. *The Algorithm Design Manual*. Springer–Verlag, 1998.
14. Barry Smyth and Mark T. Keane. Remembering to forget: A competence–preserving deletion policy for case–based reasoning systems. In *Proceedings of the 14th International Joint Conference on Artificial Intelligence*, pages 377–382, 1995.
15. Barry Smyth and Elizabeth McKenna. A portrait of case competence: Modelling the competence of case–based reasoning systems. In *Proceedings of the 4th European Workshop on Case–Based Reasoning.*, pages 208–220. Springer–Verlag, 1998.
16. D. Randall Wilson and Tony R. Martinez. Reduction techniques for exemplar-based learning algorithms. *Machine Learning*, 38(3):257–286, 2000.
17. Jun Zhu and Qiang Yang. Remembering to add: Competence–preserving case addition policies for case base maintenance. In *Proceedings of the International Joint Conference in Artificial Intelligence (IJCAI)*, pages 234–239, 1999.