On the Consistency Management of Large Case Bases: the Case for Validation

Kirsti Racine and **Qiang Yang**
School of Computing Science
Simon Fraser University
Burnaby BC V5A 1S6
Canada
Email: {kracine,qyang}@cs.sfu.ca;
Web: http://fas.sfu.ca/cs/research/groups/CBR/
Tel: 604-291-5415; Fax: 604-291-3045

Abstract

Case-based reasoning (CBR) is a practical, relatively new technology. CBR is based on the idea that new problems can often be solved by using past solutions. The basic method used to implement CBR is to build a case base of previously solved problems. These cases are then retrieved and adapted to solve new problems. Using this CBR process, a case-based system can learn incrementally and improve its performance over time.

However, a pervasive, yet relatively ignored, problem inherent in using this approach is the possible presence of inconsistencies within and among cases. These can be in the form of contradictions within the case base, possibly causing a degradation of performance efficiency, the retrieval of two conflicting solutions or no retrieval at all. Past research has only dealt with the problem superficially.

In this paper, we present an analysis of inconsistency problems arising from contradiction in a potentially large case base. We classify these problems according to their nature, and suggest validation solutions to deal with them effectively.

Introduction

Case-based reasoning is a relatively new problem solving and knowledge reuse technique in Artificial Intelligence (Kol93b). To solve a problem, a reasoner recalls previous situations **similar** to the current one and **adapts** them to help solve the current problem. The existing problem descriptions, known as *cases*, are used to suggest a means of solving the new problem, to warn the user of possible failures that have been observed in the past, and to interpret the current situation. In many practical application domains, this technique is more effective in solving problems than **rule-based expert-system** approaches, since it can overcome the so-called *knowledge acquisition bottleneck* by storing entire cases for later analysis, rather than asking the domain experts to extract their knowledge in the forms of rule-like languages. Examples of successful applications are those where extensive previous knowledge exists in recorded forms, including a help-desk system for suggesting repairs to COMPAQ printers (NCL93), and a system for helping with manufacturing design (HT95).

A pervasive, yet relatively ignored, problem inherent in using this approach is the possible addition of incorrect or incomplete cases to the case base. As the case base grows, errors within the case base become increasingly difficult to detect. The result can be contradictions or inconsistencies within a case base. These problems can potentially harm the performance of a case based reasoning system. This is because the presence of inconsistent cases will not only place a large burden on a case retriever, but also possibly present incorrect solutions to the users. These problems deserve careful study; if not handled properly, the coverage of a case based system could be badly affected, making some user queries un-answerable.

We address the case-base consistency problems in this paper. We start out by clarifying the need for the validation facility in a case-based system by presenting illustrative examples. We then present a classification of the potential problems associated with inconsistency. Finally, we propose several solutions aimed at solving these inconsistency problems under different contexts.

Problem Identification

In this section we clarify the inconsistency problems inherent in a case base. We start out by presenting an example case base for a car diagnosis domain.

An Example Domain

Consider an car diagnostic and repair example. Two cases are shown in Table 1. These cases are stored in a *case base*, which could be implemented by a database, or any legacy data source.

An important requirement for a case base reasoner is efficient retrieval. To make this possible one can identify a number of important features by which to ask a user. In this example the features (or indexes) include the make of the car, the model of the car, the year of the car, engine type and mileage, and so on.

In any real world situations it is likely that the number of features is quite large; in the car domain one can

Case 1		
make: mazda	model: 626	model year 1988
engine type: 20L EFI	mileage 12,498	
problem:	Engine is stalling	
validation procedure:	fuel injector clogged $\rightarrow$ condition of fuel injector.	
repair:	clean the MAZDA 88 fuel injector.	
Case 2		
make: Toyota	model: Camery	model year 1987
engine type: 2.8L	mileage 67,183	
problem:	No good gas mileage	
validation procedure:	had a broken gas pump $\rightarrow$ condition of gas pump.	
repair:	Replace the Toyota 87 gas pump	

Table 1: Example cases for automobile diagnosis and repair.

New Problem		
make: mazda	model: 626	model year 1985
engine type: 20L EFI	mileage 51,293	
problem:	Engine does not start	

Table 2: New problem for a given car

identify close to 30 different features. Some features such as the mileage have common-sense meanings to average users, and thus can be directly presented as questions for a user to answer. Others, such as engine type (20L EFI), may not be meaningful to an average user of the system. Thus a translation process must be in place to ensure that a higher level question is asked and its answers translated to these lower level index values. In this domain an example of a higher level question is "Is the car engine powerful?" An answer to this question could be translated to detailed specifications such as engine type, fuel type and auto-insurance rate.

One important use of the case base is in solving a new problem. Suppose that the new problems with a given car are described in Table 2.

For this task, the case-based reasoning system might perform the following operations:

1. **retrieval:** retrieve the MAZDA 88 case from case base,
2. **adaptation:** new repair action: clean the **MAZDA 85** fuel injector,
3. **learning:** decide if the MAZDA 85 case should be saved in case base, and whether the MAZDA 88 case should be removed.

Criteria for Evaluating Case Bases

There are many different ways to evaluate the quality of a case base. In this section we explore some of the criteria by which one can judge the effectiveness of a case base. Intuitively, an "effective" case base is one which is able to answer as many queries as possible efficiently and correctly.

More specifically, we evaluate a case base by the following criteria.

Consistency Consistency can be defined in many different ways. A single case may be consistent with the background knowledge, if it "makes sense" in the context of the knowledge. Similarly, two cases must be consistent with each other when both are used in a composite solution. The former is called intra-case consistency, while the latter is called inter-case consistency. In the automobile domain, a case is inconsistent with the background knowledge if an engine type is not available given the particular model of a car. In the same domain, an engine-diagnosis case is inconsistent with an exhaust-diagnosis case if they result in incorrect explanations for the problem of a car.

Correctness The correctness of a case base is measured by how often the case that is retrieved is the case in the case base that answers the query most effectively. Due to the heuristic nature of case bases, this is a difficult criterion to measure.

Redundancy Due to the ever evolving nature of case bases, it is important to have a mechanism to determine if the incoming case is subsumed by other cases in the case base or if it subsumes existing cases in the case base. If two or more cases in a case base are very similar and are retrieved for the same set of queries, it is unnecessary to keep both in the case base and may degrade the efficiency of the case reasoner.

Revision Effort The revision effort is defined as the cost associated with revising the retrieved case to answer the query.

Coverage A case base should be able to answer the full set of queries that it purports to satisfy. We call this criterion coverage as in (SK95).

Reachability Reachability is also defined as in (SK95). Given the set of cases that the reasoner purports to satisfy, the reachability of a case base can be defined as the set of cases needed to provide solutions for these problems.

Retrieval Cost The retrieval cost is measured by the number of disk accesses necessary to retrieve the correct case, given the problem description, from the case base.

Relevancy A case base should only present to the user those cases relevant to the problem at hand. For example, a problem description regarding automobile problems should not result in information about gardening.

Abstractness A case base can either contain concrete cases or it can be generalized. The level to which the case base is generalized is the "abstractness" of a case base.

Due to limited space, in this paper we only focus on the first criterion, namely consistency.

Consistency Problems

When a case base gets large, the number of inconsistent cases will inevitably increase as well. Below, we classify the consistency problems in a case base in two dimensions (see Figure 1): on the number of cases involved in a constraint violation, and on the way inconsistencies present themselves; that is, soft vs. hard constraint violations.

Intra-Case Inconsistencies Intra-case inconsistency occurs when values assigned to different features within a single case violate one or more constraints.

Consider the following example.

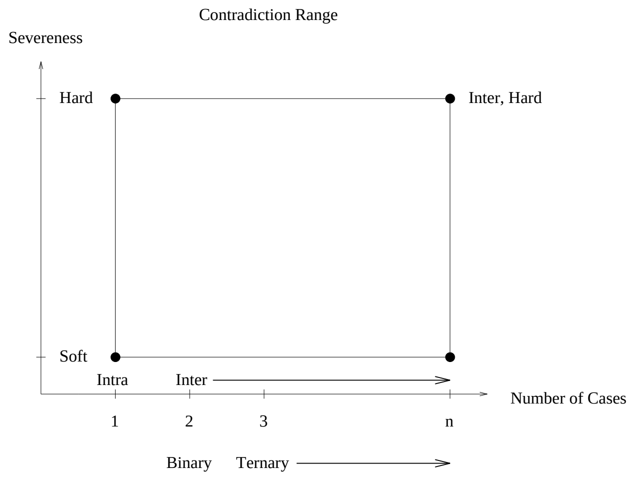

Figure 1: Dimensions of problem classification.

Attribute	Case
Type of Car	Toyota
Name of Car	Corolla
Year of Make	1995
Other Attributes	...
Transmission	Standard
Size of Interior	8 cubic meters

The above example seems totally reasonable - unfortunately, 1995 Toyota Corollas are only available with automatic transmission. Therefore, this case is self-contradictory. Cases such as this one should be identified before being entered into the case base.

The above example is an instance of a hard constraint violation. A *soft* constraint violation could also occur in the intra-case situation. As an example in the car domain, a car's description might specify its mileage with an uncommonly high value (say, 100,000 miles). In this case, a warning is desired to bring this item to the user's attention.

Inter-case Inconsistencies More difficult to deal with are the inter-case consistency violations. Inter-case contradictions are those which occur across two or more distinct cases. Consider the example in Table 3.

The reason that the second case proposes such a dire solution is that the car in that case is a *1987* Honda. Let us suppose that '87 Hondas have structural problems causing engine leaks in some cases and that fluctuating power is a symptom of these leaks. If the valve is replaced while the engine is leaking, the person changing the valve could suffer serious irritation and inflam-

Attribute	**Case 1**	**Case 2**	**Case 3**
Type of Car	Honda	Honda	Honda
Type of Engine	8 Cylinder	8 Cylinder	8 Cylinder
Year of Make	1994	?	1994
More Attributes	...	...	...
PROBLEM	Fluctuating Power	Fluctuating Power	Fluctuating Power
SOLUTION	Replace valve	DO NOT TOUCH CAR!	DO NOT replace valve

Table 3: Inter-case contradiction examples.

mation. Due to noise, the attribute *Year of Make* was not recorded. In this case, it would be useful to signal a contradiction so that a human expert can determine why the solutions appear to oppose one another.

Similarly, consider the situation where the attribute *Year of Make* is entered incorrectly as *1994* - in this case we have a *hard* contradiction; two cases match on every attribute, yet propose conflicting solutions. Again, if a contradiction signal is shown at the time the second case is being entered into the case base, this error can be detected and corrected.

Table 3 illustrated some possible instances of inter-case contradictions. Finding the contradiction between the first and the second case involves multiple steps. The two cases do conflict with each other, but the system must know that replacing the valve involves touching the car to identify this contradiction. The conflict between the first and third cases is more obvious - it is a one step contradiction. These examples were provided to both prove the need for a validation mechanism and the problems inherent in implementing one.

There are more types of possible contradictions, two of which are subset and temporal contradictions. A subset contradiction occurs when a solution in a case only refers to a subset of the queries that may access the case. The problem is that the necessary discriminator is not an attribute of the case. A temporal contradiction would occur when the correct solution changes over time. Table 4 illustrates both types of contradictions.

In Table 4, both cases are valid. The reason that the two cases offer different solutions is that in the first case, the time elapsed between the car overheating and the user querying the case base was sufficient time for the car to have cooled down. If coolant is added while the car is still heated, it will spray out of the tank and possibly harm the applicant. However, no time

Attribute	**Case 1**	**Case 2**
Type of Car	Toyota	Toyota
Name of Car	Corolla	Corolla
Year	1994	1994
More Attributes	...	...
PROBLEM	Overheating	Overheating
SOLUTION	Add Coolant	Do nothing

Table 4: Inter-case contradiction examples.

attribute is available in the case base. To solve problems of this sort, it is necessary to add the necessary discriminators.

Previous Research

There has been very little research done specifically on maintaining the consistency of a case base. Many of the land mark books on case-based reasoning suggest that it is an area which needs further attention, (Kol93a), (SKR94). These texts also offer many suggestions why it is necessary to maintain the integrity of a case base including maintaining competence of the system, the possibility of returning contradictory solutions and the possibility of returning no solution whatsoever to a query that should be answered.

Most of the research in this area is concerned with *optimization*. Due to the large size of some case bases, it is necessary to "forget" cases as time goes by or retrieval stages become increasingly expensive (SK95). The strategy of deciding which cases to forget is similar to the question of validation. Some researchers advocate a random deletion policy (MS88). This is a very simple, inexpensive policy and is completely domain independent. Simply randomly select and delete a case from the case base. A slightly more complicated approach is to calculate the frequency that each case is retrieved and delete those who are not frequently accessed (Min90) [1]. The problem with both of these approaches is that "important" cases can be deleted. In other words, a case that is necessary to answer a query or set of queries can be deleted from the system

To overcome this problem, Smyth et. al (SK95), suggested a competence-preserving deletion approach. The premise of this approach is that each case in the base should be classified according to its competence. These classifications are made according to two key concepts: *coverage* and *reachability*. Coverage refers to the set of problems that each case can solve. Reachability is the set of cases that can be used to provide solutions for each current problem. Cases that are the only case that can answer a specific query are *pivotal* cases. *Auxiliary* cases are those which are completely subsumed by other cases in the base. In between these two extremes are the *spanning* cases which link together areas independently covered by other cases and *support* cases which exist in groups that support an idea. The deletion algorithm then deletes cases in the order of their classifications : auxiliary, support, spanning and then pivotal cases.

Smythe has also written a paper on incremental case-based reasoning in which he advocates classifying cases through the use of a decision tree (SC95). A decision tree is a classification mechanism. Each branch of the tree corresponds to a different class of cases. Although the paper does not address the problem of validation, this approach could be modified to perform validation through optimization. Classifications would obey the constraints of the system, thereby eliminating the more obvious contradictions. Using the decision tree approach, discriminators are identified to distinguish different classes. These discriminators can then be added to the system in the form of additional constraints. This can potentially solve the subset contradiction problem. Also, this approach does address the problem of missing data in cases. Through the use of induction, this data is discovered by extrapolating data from similar cases.

The approaches mentioned above are motivated by the need to delete cases in order to maintain the case base at a reasonable size. However, the focus of this paper is to establish an approach that can identify incorrect, incomplete or inconsistent cases as they are added to the case base regardless of the current size of the case base. Very few papers address this problem directly. One of the papers that does address this problem focuses on noisy cases, specifically cases that have incorrect or incomplete information (Sha91). This paper refers to a case based reasoner to identify genes. Due to the nature of the domain of this system, incorrect and incomplete information is difficult to detect by humans. The errors typically take the form of extraneous or missing DNA strands which render the cases incorrect and unusable. The approach used to overcome these errors involves generating all possible partial matches to the current case and then combining them to achieve a global picture.

[1] Ironically, this policy degrades the competence of the case base more than the random deletion policy (SK95)

Another approach to extrapolating incomplete data has been suggested by Simoudis, (Sim92). He has done extensive research on a process named *validated retrieval*. Validated retrieval is very similar to the above process of generating all possible matches, but uses heuristics to reduce the processing time. As each query is executed, all similar cases are located. Validated retrieval uses only these similar cases to extrapolate missing information, thereby reducing the number of cases to be considered.

The papers above provide the needed background knowledge to implement a validation mechanism for a case-based reasoner. However, none of them specifically address the current problem. A validation mechanism must identify inconsistent cases a high percentage of the time. Furthermore, it should not significantly increase processing time. Possible approaches are discussed in the next section.

Proposed Solutions

We are currently investigating different ways to handle the different types of contradictions mentioned above. We aim at devising a validation module which could handle the following tasks:

Inconsistency Detection A system for detecting soft or hard inconsistency will be devised. The system will present warnings to a user should inter or intra case inconsistency occurs.

Inconsistency Correction When detected, a facility should be provided to the user for correcting these inconsistencies. This could range from interactive correction methods to automatic ones.

Inconsistency Prevention An attractive approach would be to prevent the contradiction from happening in the first place. This could be accomplished by a rule based system, or a truth-maintenance system, which could make inferences on the range of potential values to questions, based on a subset of answers obtained so far.

Optimization of Validation Methods To make it efficient for the detection and correction methods to work, it is necessary to reduce the number of consistency rules and the number of cases under scrutiny. This is referred to as optimization.

We are currently developing two solutions to the validation problem. One solution is to use a forward-chaining rule based system for validation. The rules could record integrity constraints for maintaining consistency in the case base. For example, one rule could state that for all cars, there is only one engine; furthermore, the engine must be a car engine. If this rule is applied to all incoming cases, the intra-case contradiction mentioned above could be detected. Another example of a constraint is

If $type\_of\_car(Toyota) \wedge name\_of\_car(Corolla) \wedge year\_of\_make(1995)$
Then $transmission(automatic)$.

Such constraints could be enforced in different ways. A warning could be issued as soon as an invalid field is entered or as each field is entered values could be enforced in the following attributes. In this case, as soon as the user identified the car as a '95 Toyota Corolla, the system would fill in the transmission attribute as standard. The rules could also be generalized to handle groups of cases. Therefore the same approach could be applied to inter-case consistency problems, for dealing with both soft and hard constraint problems.

There are a number of problems with the rule-based approach. The first problem is the knowledge-acquisition bottleneck. The primary advantage of using case-based reasoning is that domain experts are not required to lend their expertise in the form of rules. To address this problem, our research is focused on both reducing the number of rules necessary to implement our validation mechanism **or** discovering the rules implicitly stored in the case base as discussed in (HF96). The latter approach alleviates the necessity of obtaining information from a domain expert. Given a rich enough case base, this approach should return strong enough rules to solve many contradiction problems.

A further problem is the number of cases to which the rules must be applied may be very large. A large case base could therefore be very inefficient to check for consistency. To solve this problem, we could supply a concept hierarchy for features, whereby a case can be abstracted at various levels of abstraction in this hierarchy. This can be done for a given case by replacing the concrete values for indexes by the corresponding concept symbols in the hierarchy. A similar idea was explored in (HF96) for mining conceptual rules in a database. The effect of this substitution process is that the case base would be collapsed to a smaller size. At the same time the rules could also be abstracted to contain the same language terms. When the abstracted rules are applied to the abstracted database, the process of consistency-checking is expected to be much more efficient.

An obvious advantage in using the concept hierarchy approach is that many soft constraint violations such as range aberrations can be detected easily . If an attribute can not be generalized using the concept hierarchy, the value for that attribute is likely out of range or incorrect. Also, using this approach effectively optimizes the case base. Hopefully, the abstracted case base can be used to answer a large proportion of the queries enabling efficient retrieval. Case bases are continually growing; developing an abstract representative case base can assist in query answering efficiency.

One disadvantage of this approach is the reliance on background knowledge. Often concept hierarchies can be discovered from existing data (HF96). However, attributes that have too many distinct values can not be generalized in this fashion. Also, difficulties arise in breaking up continuous valued attributes into distinct intervals. For some domains, such as DNA identification, this approach may be untenable.

Currently, our research is directed toward testing this method on a variety of domains.

Conclusions

In this paper we have accomplished three objectives: (1) we have clarified the need for validation in a case based reasoning (CBR) system; (2) we have classified different types of consistency problems into classes, so that a divide-and-conquer approach can be applied to solve these problems; and finally, (3) we have pointed out a logic-based method for validating a case base.

Acknowledgment

We wish to thank the following colleagues —- from SFU: D. Edward Kim and Philip W.L. Fong; from MPR Teltech Ltd.: Jose Lee, Nick Duncan and Wendy Moore.

The authors are supported by grants from: Natural Sciences and Engineering Research Council of Canada (NSERC), MPR Teltech Ltd., Science Council of BC , BC Advanced Systems Institute , The Simon Fraser University , Ebco Industries Ltd. , Epic Data International , MacDonald Dettwiler and Associates , Glenayre Electronics Inc. , BC Telephone Company and Ontario Centers of Excellence (ITRC).

References

Jiawei Han and Yongjian Fu. Exploration of the power of attibute-oriented induction in data mining. In Fayyad, Piatetsky-Shapiro, Smyth, and Uthurusamy, editors, *Advances in Knowledge Discovery and Data Mining*, chapter 4, pages 399–424. AAAI Press/MIT Press, 1996.

David Hinkle and Christopher Toomey. Applying case-based reasoning to manufacturing. *AI Magazine*, (Spring):65–73, 1995.

J. Kolodner. *Case-Based Reasoning.* Morgan Kaufman, California, 1993.

Janet Kolodner. *Case-based Reasoning.* Morgan Kaufmann Publishers, Inc., San Mateo, CA, 1993.

S. Minton. Qualitative results concerning the utility of explanation-based learning. *Artificial Intelligence*, 42:363–391, 1990.

S. Markovich and P. Scott. The role of forgetting in learning. *Proceedings of the Fifth International Conference on Machine Learning*, 5:459–465, 1988.

Trung Nguyen, Mary Czerwinski, and Dan Lee. Compaq quicksource—providing the consumer with the power of ai. *AI Magazine*, 1993.

B. Smythe and P. Cunningham. A comparison of incremental case-based reasoning and inductive learning. *Proceedings of the 2nd European Workshop on Case-Based Reasoning*, 2, 1995.

J. Shavlik. Finding genes by case-based reasoning in the presence of noisy case boundaries. *Proceedings of the 1991 DARPA Workshop on Case-Based Reasoning*, 1:291–303, 1991.

E. Simoudis. Using case-based retrieval for customer technical support. *IEEE Expert*, 7(5):7–13, 1992.

B. Smythe and M. Keane. Remembering to forget : A competence-preserving case deletion policy for case-based reasoning systems. *International Joint Conference on Artificial Intelligence*, 1:0–0, 1995.

R. Shank, A. Kass, and C. Riesbeck. *Inside Case-Based Reasoning.* Lawrence Erlbaum Associates, New Jersey, 1994.