Remembering To Forget
A Competence-Preserving Case Deletion Policy for Case-Based Reasoning Systems

Barry Smyth
Hitachi Dublin Laboratory,
Trinity College Dublin,
Dublin, IRELAND.
EMail: barry.smyth@hdl.ie

Mark T. Keane
Department of Computer Science,
Trinity College Dublin,
Dublin, IRELAND.
EMail: mark.keane@cs.tcd.ie

Abstract

The *utility problem* occurs when the cost associated with searching for relevant knowledge outweighs the benefit of applying this knowledge. One common machine learning strategy for coping with this problem ensures that stored knowledge is genuinely useful, deleting any structures that do not contribute to performance in a positive sense, and essentially limiting the size of the knowledge-base. We will examine this *deletion* strategy in the context of case-based reasoning (CBR) systems. In CBR the impact of the utility problem is very much dependant on the size and growth of the case-base; larger case-bases mean more expensive retrieval stages, an expensive overhead in CBR systems. Traditional deletion strategies will keep performance in check (and thereby control the classical utility problem) but they may cause problems for CBR system competence. This effect is demonstrated experimentally and in reply two new deletion strategies are proposed that can take both competence and performance into consideration during deletion.

1 Introduction

The traditional wisdom in knowledge-based systems has been that more knowledge is a good thing. However, in recent years this view has been questioned with demonstrations that certain "harmful" knowledge may actually degrade system performance (usually performance is taken as problem solving efficiency or time). This problem has been dubbed the utility problem [Minton, 1990; Tambe *et al,* 1990]. In case-based reasoning systems (CBR) a special case of the utility problem arises called the swamping problem fFrancis & Ram, 1993b]. The swamping problem relates to the expense of searching large case-bases for appropriate cases with which to solve the current problem.

One could argue that efficient parallel retrieval algorithms are the solution to the swamping problem; in other words by keeping retrieval time constant, increasing the case-base size no longer threatens system efficiency. While this is true to a degree, it is not currently realistic, and until massively parallel machines become the norm other strategies must be used; in fact, while massively parallel retrieval algorithms do help in delaying performance degradation, they do not eliminate it altogether.

Markovitch and Scott [1993] have characterised different strategies for dealing with the utility problem in terms of information filters applied at different stages in the problem solving process (e.g., acquisition, feature extraction, matching, learning). Two common strategies which are especially relevant to the swamping problem are selective utilization and selective retention. Selective utilization filters [Markovitch & Scott, 1989] prevent the problem solver from "seeing" certain knowledge items for some period of time, and are often implemented in CBR systems as indexing schemes [Kolodner, 1994] or by time bounding the case-retrieval mechanism [Brown, 1993; Veloso, 1992]. In these approaches only a subset of the case-base is made available to the retrieval mechanism with the attendant disadvantage that certain cases may be missed at retrieval time. Unfortunately, these omissions can result in unnecessarily complex adaptation stages or even problem-solving failures. As an alternative, selective retention filters permanently remove knowledge from the knowledge-base [Holland, 1986; Markovitch & Scott, 1988; Minton, 1990] and are represented in CBR systems by techniques that only learn certain cases or that delete cases that appear to be redundant; the selection of a case to delete is controlled by a deletion policy. An advantage with selective retention is that the whole case-base is available at retrieval time but it is critical that only high quality cases are stored to keep the case-base small and retrieval time at acceptable levels.

In this paper, we investigate a number of traditional deletion policies and their application to CBR. In section 2, we argue that such policies may not be directly applicable to CBR systems because they damage the competence potentials of such systems over time. Section 3 introduces two new policies designed specifically with CBR in mind. These new policies recognise the possibility of competence degradation through case deletion and safe-guard against it by using an explicit model of case competence to guide deletion. Finally, in section 4, before a concluding look at the applicability and appropriateness of our approach, we demonstrate its effectiveness with the aid of experimental results.

2 Coping With The Utility Problem

Machine learning research offers a variety of different deletion policies, most of which should work well to preserve performance in CBR systems (and hence limit the classical utility problem). However, they can result in a gradual but irreversible reduction in system competence. In short, to properly control the CBR utility problem, a deletion policy is required that preserves both performance and competence.

2.1 Traditional Deletion Policies

A simple deletion policy is *random deletion.* According to this policy a random item is removed from the knowledge-base once the knowledge-base size exceeds some predefined limit. Surprisingly, random deletion can work very well and can often be as effective as more principled (and expensive) methods [Markovitch and Scott, 1988].

One such more principled approach is Minton's utility metric [Minton, 1990] –

Utility=(ApplicationFreq*AverageSavings)-MatchCost

- which chooses a knowledge item for deletion based on an estimate of its performance benefits. This *utility deletion* policy removes knowledge items with negative utility. Although it is not necessary to directly limit the size of the knowledge-base, the frequency factor tends to indirectly control size because as the knowledge-base grows the application frequency of a particular item tends to drop and so its utility estimate degrades. By using this policy substantial performance improvements were observed in the Prodigy/EBL system [Minton, 1990]. Similar policies have been employed elsewhere with equally successful results [Markovitch and Scott, 1993; Markovitch and Scott, 1988; Keller, 1987].

2.2 Problems with Classical Deletion Policies

CBR systems are prone to utility problems (especially swamping problems) in much the same way as other problem solvers. The hope is that deletion policies such as the above will transfer well to case-based systems. However, there is one important difference between pure CBR systems and the speed-up learning systems (epitomised by Prodigy/EBL and SOAR) where random and utility based deletion have been successful. Pure CBR systems do not have a first-principles problem solver; that is, without cases, CBR systems cannot solve problems at all. In contrast, the case-like knowledge of systems such as Prodigy/EBL and SOAR serves only as speed-up knowledge. Therefore, it is perfectly fine for speed-up learners to use performance related policies, like Minton's utility metric, because no matter what knowledge is deleted, the system's competence is not altered; problems can always be solved from first-principles. However, these deletion policies can have disastrous results for case-based reasoners. The deletion of critical cases can significantly reduce the competence of a CBR system, rendering certain classes of target problems permanently unsolvable.

In CBR, all cases are not equal. Some cases contribute mainly to the competence of the system and others may predominantly contribute to its performance. *Pivotal cases* empower the system with its basic competence. *Auxiliary cases,* on the other hand, only contribute to performance. The deletion of a pivotal case results in an irreversible reduction in the competence a CBR system. Traditional deletion policies are not sensitive to these different categories of cases. Consequently, such techniques can delete pivotal cases if they do not contribute to performance. The result is an irreversible drop in competence.

3 Competence-Preserving Deletion

In this section, we describe how the competence of a CBR system can be modelled and how deletion policies can exploit this model to guard against competence depletion while controlling the size of the case-base in a manner that guards against the swamping problem.

3.1 Case Competence Categories

We have found it useful to consider four basic classes of cases. First, there are pivotal and auxiliary cases that represent the extremes of our competence model. In addition, there are intermediate categories of *spanning* and *support* cases that correspond to cases whose deletion may or may not reduce competence depending on what other cases remain in the case-base. By categorising cases according to whether they are pivots, spanning, supporting or auxiliary, it is possible to obtain a picture of the case-related competence of a given system.

Coverage and Reachability

The key concepts in categorising cases are *coverage* and *reachability.* The coverage of a case is the set of target problems that it can be used to solve. The reachability of a target problem is the set of cases that can be used to provide a solution for the target. Obviously, computing these sets for every case and target is impossible; the space of target problems is, in general, simply too vast.

A more tractable solution is to assume that the case-base itself is a sample of the underlying distribution of target problems. Now, we can estimate the coverage of a case by the set of *cases* that can be solved by its retrieval and adaptation, and the reachability of a case by the set of *cases* that can bring about its solution (see Definition 1 and 2 respectively).

Definition 1: Coverage

Given a case - base $C = \{c_1, \ldots, c_n\}$, For $c \in C$,

$\text{Coverage}(c) = \{c' \in C : \text{Adaptable}(c, c')\}$

Definition 2: Reachability

Given a case - base $C = \{c_1, \ldots, c_n\}$, For $c \in C$,

$\text{Reachable}(c) = \{c' \in C : \text{Adaptable}(c', c)\}$

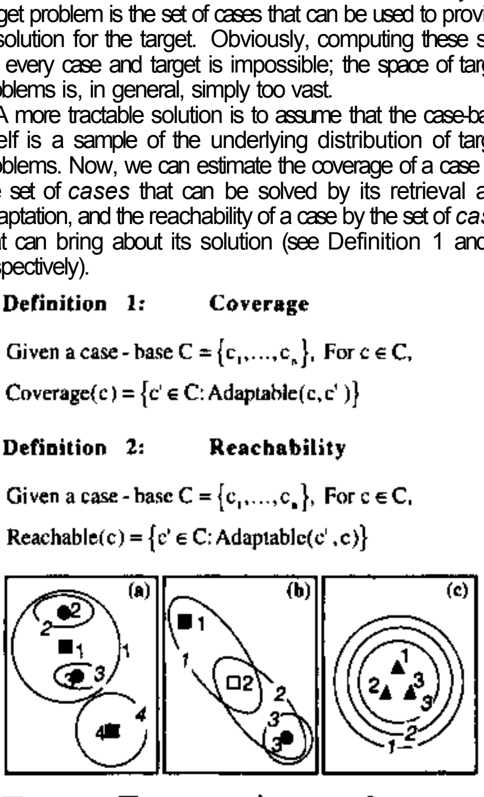

Figure 1. Case Categories

Figure 1 illustrates the different case categories in terms of their coverage (and reachability) sets. Each case is numbered and its conesponding coverage set is labelled with the same number; for example, in Figure 1(b), Coverage(2)={2,3) and Reachable(2)={ 1,2). We will refer to these configurations in the coming sections.

Pivotal Cases

A case is a pivotal case if its deletion directly reduces the competence of a system (irrespective of the other cases in the case-base). Using our above estimates of coverage and reachability a case is pivotal if it is reachable by no other case but itself (Definition 3).

Definition 3: Pivotal Case

$$\text{Pivot}(c) \text{ iff } \text{Reachable}(c) - \{c\} = \phi$$

Pivotal cases are generally outliers, being too isolated to be solved by any other case. Consequently, target problems falling within the region of pivot can only be solved by that pivot. For example, in Figure 2(a) two pivotal cases are shown, cases 1 and 4. It is clear that should either be deleted then at least two target problems cannot be solved, namely the problems corresponding to case 1 and 4.

Auxiliary Cases

Auxiliary cases do not effect competence at all. Their deletion only reduces the efficiency of the system. A case is an auxiliary case if the coverage it provides is subsumed by the coverage of one of its reachable cases (Definition 4).

Definition 4: Auxiliary Case

$$\text{Auxiliary}(c) \text{ iff } \exists c' \in \text{Reachable}(c) - \{c\}:$$
$$\text{Coverage}(c) \subset_{*} \text{Coverage}(c')$$

Auxiliary cases tend to lie within clusters of cases. If one is deleted then a nearby case can be used to solve any target that the deleted auxiliary could solve, so competence is not reduced. Figure 2(a) has two auxiliary cases, 2 and 3. The coverage of each is a proper subset of the coverage offered by case 1 and therefore any target problem that case be solved by cases 2 or 3 can also be solved by case 1.

Spanning Cases

Spanning cases do not directly affect competence. They are so named because their coverage spaces link (or span) regions of the problem space that are independently covered by other cases (see Definition 5). If cases from these linked regions are deleted then the spanning case may be necessary.

In Figure 2(b) case 2 is a spanning case with its coverage space joining case 1 and case 3 but offering no more coverage that 1 and 3 provide together. However it is important to realise that if case 3 is deleted then the spanning case is now necessary.

Definition 5: Spanning Case

$$\text{Spanning}(c) \text{ iff } \neg\text{Pivotal}(c) \wedge \text{Coverage}(c)$$
$$\cap \bigcup_{c' \in \text{Reachable}(c) - \{c\}} \text{Coverage}(c) \neq \varnothing$$

Support Cases

Support cases are a special class of spanning cases and again do not affect competence directly. They exist in groups, each support providing similar coverage as the others in a group. While the deletion of any one case (or indeed any proper subset) of a support group does not reduce competence, the removal of the group as a whole is analogous to deleting a pivot, and does reduce competence. More formally, the definition of support cases is given below in Definition 6.

Definition 6: Support Case

$$\text{Support}(c) \text{ iff } \exists c' \in \text{Reachable}(c) - \{c\}:$$
$$\text{Coverage}(c') \subset \text{Coverage}(c)$$

Furthermore, a set of cases C form a support group if they all provide the same coverage (Definition 7).

Definition 7: Support Group

$$\text{SupportGroup}(C') \text{ iff } \forall c_i, c_j \in C':$$
$$\text{Coverage}(c_i) = \text{Coverage}(c_j)$$

Figure 2(c) shows a support group of three support cases. The removal of any two of these cases does not influence competence, the coverage offered by the third being equivalent to that offered by the deleted two.

3.2 Modelling Case Competence

While computing the competence categories may be expensive they are only computed once as a start-up cost. Strictly speaking, during future problem solving as cases are learned and deleted from the case-base, the case categories must be updated by recomputing the coverage and reachability sets of the appropriate cases and adjusting the categories accordingly. This is obviously too expensive to perform every time learning and deletion occurs. Instead, we propose the following heuristics for efficiently updating the case categories. Algorithm 1 outlines how the competence categories change as new cases are learned.

```
Learning Update( target,base):
If Pivotal(base) then
    Remove the base from the set of pivotal cases
    Add the base and target as a new support group
    and mark this group as pivotal in origin
ElseIf Support(base) then
    Add the target to the base's support group
ElseIf Spanning(base) then
    Make new support group from the base & target
      and mark this group as spanning in origin
ElseIf Auxiliary(base) then
    Add the target to the set of auxiliary cases
EndIf
```

Algorithm 1. Learning Update

Algorithm 2 describes what happens on deleting a case from the case-base.

```
DeletionUpdate(case):
If Pivotal(case) then
  Remove the case from the set of pivotal cases
Elseif Spanning(case) then
  Remove the case from the set of spanning cases
ElseIf Support(case) then
  Remove the case from its support group
  If the resulting group is a singleton then
    If the group is pivotal in origin then
       Add this remaining case to the pivots
    ElseIf add this case to the spanning set
       Remove singleton from the support groups
    EndIf
ElseIf Auxiliary(case) then
  Remove the case from the set of auxiliary cases
EndIf
```

Algorithm 2. Deletion Update

Note that all of these procedures are estimates, they make the assumption that the space of target problems is accurately approximated by the case-base. If this is true then, as our experiments will show, an accurate picture of competence can be maintained and our new deletion policies work well. If it is not true then the effectiveness of these policies will deteriorate.

3.3 The Footprint Deletion Policy

Ideally a deletion policy should work to remove irrelevant cases thereby guiding the case-base towards an optimal configuration of cases (optimal in the sense that it maximises competence while minimising size). We call this optimal case-base the *competence footprint,* the competence footprint provides the same competence as the entire case-base but with fewer cases.

The case categories described above provide a means of ordering cases for deletion in terms of their competence contributions. The auxiliary cases are the least important as they make no direct contribution to competence, next are the support cases, then the spanning cases, and finally the pivotal cases. This is the basis of the first of our new policies; as outlined in the algorithm below (Algorithm 3) auxiliary cases are selected for deletion before support cases, which in turn are chosen before spanning and pivotal cases.

This is the basic *footprint deletion* (FD) strategy. In addition, further sub-strategies must be formulated. For example: Which auxiliary case is chosen from the set of auxiliary cases?; Given a number of equal sized support groups, which group is chosen and which case from this group is selected; Finally, if a pivot must be deleted, then which one? One approach is to choose the candidate with the largest reachable set. This would mean that the case chosen is the one which can be solved by the greatest number of existing cases, thus limiting the impact of the deletion on the real system competence. Another, approach is to choose the case with the least coverage.

```
DeleteCase(Cases):
 If there are auxiliary cases then
   SelectAuxiliary(AuxiliaryCases)
 ElseIf there are support cases then
   With the largest support group
      SelectSupport(SupportGroup)
 ElseIf there are spanning cases then
   SelectSpanning(SpanningCases)
 ElseIf there are pivotal cases then
   SelectPivot(PivotalCases)
 EndIf
```

Algorithm 3. The Footprint Deletion Algorithm

3.4 Combining Competence & Performance

Footprint deletion is not designed to eliminate the need for performance-based methods such as utility deletion. Since it is only designed to consider competence we still need to deal with the performance aspect. Indeed, one interesting use of footprint deletion is in collaboration with existing performance-based policies. For example, in partnership, footprint and utility based policies can constrain deletion to preserve both competence and performance.

Combining footprint deletion and utility deletion is very simple; we will refer to the resulting hybrid strategy as *footprint-utility deletion (FUD).* First, the footprint method is used to select candidates for deletion. If there is only one such candidate then it is deleted. If, however, there a number of candidates, then rather than selecting the one with the least coverage or largest reachability set, the candidate with the lowest utility is chosen. In other words the utility metric is used within the SclectPivot, SelectSpanning, SelectSupport, and SelectAuxiliary procedures referred to above; in this way utility estimates can be used to select between a number of alternative auxiliary cases or between a number of support cases belonging to the same support group or between a number of pivotal cases.

In fact, a disadvantage of using footprint deletion on its own is that low utility cases may be preserved while high utility cases with the same competence contributions may be deleted. By combining footprint deletion and utility deletion this problem is immediately solved because low utility cases will always be deleted instead of high utility cases with the same competence characteristics.

4 Experimental Analysis

Our approach to controlling the swamping problem is to restrict the size of the case-base to ensure retrieval costs are kept within acceptable limits. That this strategy controls swamping should be clear. Our objective in the following experiments is to provide empirical support for two claims.

First, that traditional deletion methods *are* subject to competence degradation whereas as the footprint policies are not. This is investigated in Experiment 1 by imposing a swamping limit on the case-base and monitoring the competence of the system as different cases are deleted according to different policies.

Second, that the footprint policies work towards a case-base that maximises competence while minimising size whereas the random and utility methods do not. This is investigated in Experiment 2 by varying the swamping limit imposed on the case-base and again monitoring how systems competence develops.

The experiments were run on a simple CBR system for residential property valuation using a simple nearest-neighbour retrieval algorithm; this domain has been introduced in Smyth & Cunningham [1995].

4.1 Experiment 1

An initial case-base of 50 cases and a set of 160 target problems were formulated. First, to test the basic competence of the case-base the target problems were presented to the system with no learning taking place. The result was a benchmark competence of 100% for the given case-base and problem set; that is, the test case-base was capable of solving all of the target problems. In the next studies, learning was allowed but the size of the case-base was restricted to 75 cases, the *swamping limit.*

Four studies were carried out by using the same target problems but by varying the deletion policy. During each run, learning proceeded unhindered until the case-base size reached the swamping limit. Subsequent learning could only take place after a case had been removed from the case-base.

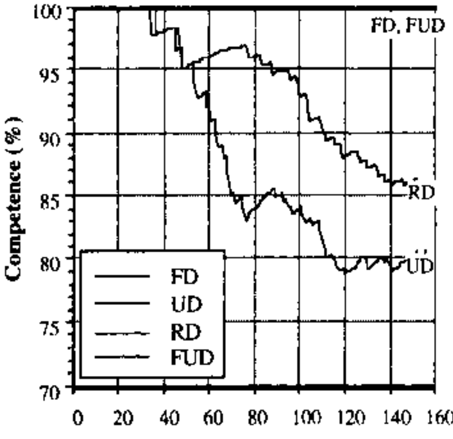

Figure 2. Competence Depletion Results

Figure 2 shows the results obtained. The first thing to notice is that the two competence preserving policies, footprint deletion (FD) and footprint-utility deletion (FUD) managed to maintain overall competence at its benchmark level of 100%. In contrast, as predicted, the random deletion (RD) and utility deletion (UD) policies resulted in significant drops in competence as important cases were deleted over the course of the experiment; almost immediately on reaching the swamping limit (after 25 trials) competence begins to drop as crucial cases are removed, and their absence renders certain target problms unsolvable. In the end overall competence had fallen to 80% and 86% of its former level for the utility and random deletion policies respectively.

4.2 Experiment 2

In CBR systems it is generally desirable to optimise the competence provided by the case-base. That is, a deletion policy should maximise the competence of the case-base while minimising its size. So it is not enough that the deletion policy simply guard against the deletion of cases that contribute to competence. It is also necessary to ensure that if a competence contributing case must be deleted, that this case is carefully selected to minimise the inevitable competence loss; alternatively, if a case can be added to the case-base then the one chosen should provide the greatest competence contribution. The four different deletion policies are tested for this property in the following experiment.

The previous experiment were re-run a number of times with a different swamping limit imposed each time and the results are shown in Figure 3. The swamping limit is varied from 5 cases to 95 cases. After each re-run the overall competence is noted and graphed.

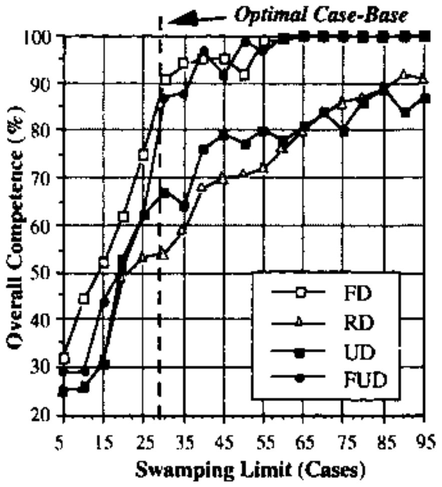

Figure 3. Swamping Limit Results

Clearly the two footprint policies are optimising competence whereas the traditional policies are not; the curves of the former rise to the maximum benchmark level much more rapidly that the curves of the latter. This means that as more and more cases are allowed into the case-base, the two footprint policies seek out those which make the largest competence contributions. In contrast, because the traditional random and utility based methods have no understanding of competence, a competence contributing case is only stored in the case-base if it is beneficial from the performance viewpoint.

In fact, the qualitative nature of the footprint deletion and footprint-utility deletion results can be explained in terms of the distribution of pivotal, spanning, support, and auxiliary cases in the case-base. Table 1 shows this distribution in the property valuation case-base. Note that the 25 support cases formed 7 support groups and there were no spanning cases.

Pivotal Cases	Auxiliary Cases	Support Cases	Spanning Cases
20	5	25	0

Table 1. Case-Base Statistics.

For the residential property cases an optimal case-base can be constructed from all the pivotal cases plus one case from each support group; that is, to achieve benchmark competence the case-base must contain at least 27 cases (all of the pivots plus 7 of the supports, one from each group). So one would expect that if the ideal deletion policy (ideal from the point of view of preserving competence) constrains the case-base towards this optimal competence configuration, then competence should reach the benchmark maximum once the swamping limit facilitates a case-base of at least 27 cases. This is what happens in the above experiment. In Figure 3, both footprint deletion and footprint-utility deletion reach about 90% of the benchmark competence once the swamping limit is just over 25 cases, whereas the random and utility policies only reach about 60% competence at this same swamping limit.

5 Conclusion

This paper demonstrated that while traditional deletion policies are effective in controlling the swamping problem from a performance perspective, they may lead to competence degradation in many CBR systems. The solution proposed uses a model of case competence to guide the learning and deletion of cases. We outlined a suite of algorithms for constructing and efficiently maintaining this competence model at runtime, and two new deletion policies (footprint deletion and footprint-utility deletion) were presented which can preserve competence by referring to this model at deletion time. The preliminary experimental results are promising in demonstrating that the competence estimates did prove useful in preserving the actual system competence. In particular, the hybrid footprint-utility deletion method should enjoy the same performance advantages as the traditional utility methods while at the same time guarding against competence degradation.

The main assumption on which the new policies are based is that the case distribution is a good representation of the target distribution. If the case distribution is not a reliable estimate of the target distribution then the competence model will not be reliable. Consequently, the effectiveness of the competence deletion policy will be limited. It is our contention that, while not every domain may be characterised in this way, many can and by exploiting this property we can safe-guard against the utility problem while at the same time avoiding the possibility of competence degradation.

Our competence modelling approach may also be used during the initial case acquisition stage of system development. It is often undesirable to store every available case in the initial case-base; for one thing there is the utility problem and secondly irrelevant cases may introduce noise into the retrieval stage and lead to the selection of sub-optimal cases or difficulties in tuning the similarity metric. By modelling the competence of the available casts an optimal initial case-base can be formed, providing a more manageable source of problem solving expertise.

References

[Brown, 1993] M. Brown. A Memory-Model for Case Retrieval by Activation Passing. Ph.D Thesis (UMCS-94-2-1). University of Manchester, United Kingdom, 1993.

[Francis and Ram, 1993a] A. G. Francis and A. Ram. Computational Models of the Utility Problem and their Application to a Utility Analysis of Case-based Reasoning. In *Proceedings of the Workshop on Knowledge Compilation and Speed-Up Learning,* 1993.

[Francis and Ram, 1993b] A. G. Francis and A. Ram. The Utility Problem in Case-Based Reasoning. Technical Report (ER-93-08). Georgia Institute of Technology, USA, 1993.

[Keller, 1987] M. R. Keller. Concept Learning in Context. In *Proceedings of the Fourth International Workshop on Machine Learning,* pages 482-487. Morgan Kaufmann, 1987.

(Kolodner, 1994] Kolodner, J. The Case Library: Representing and Indexing Cases. *Case-Based Reasoning,* pages 141-282. Morgan-Kaufmann.

[Markovitch and Scott, 1988] S. Markovitch and P. D. Scott. The Role of Forgetting in Learning. In *Proceedings of the Fifth International Conference on Machine Learning ,* pages 459-465, 1988.

[Markovitch and Scott, 1989] S. Markovitch and P. D. Scott. Utilization Filtering: A Method For reducing The Inherent Harmfulness of Deductively Learned Knowledge. In *Proceedings of the International Joint Conference on Artificial Intelligence,* pages 738-743, 1989.

[Markovitch and Scott, 1993] S. Markovitch and P. D. Scott. Information Filtering. Selection Mechanisms in Learning Systems. *Machine Learning,* 10: 113-151, 1993.

[Minton, 1990] S. Minton. Qualitative Results Concerning the Utility of Explanation-Based Learning. *Artificial Intelligence,* 42:363-391,1990.

[Smyth & Cunningham, 19951 B. Smyth and P. Cunningham. A Comparison of Incremental Case-Based Reasoning and Inductive Learning. *Topics in Case-Based Reasoning: Proceedings of the 2nd European Workshop on Case-Based Reasoning.* Springer-Verlag, 1995.

[Tambe *et al,* 1990] N. Tambe, A. Newell, and P. S. Rosenbloom. The Problem of Expensive Chunks and is Solution by Restricting Expressiveness. *Machine Learning,* 5: 299-349, 1990.

[Veloso, 1992] M. Veloso. Learning by Analogical Reasoning in general Problem Solving. Ph.D Thesis (CMU-CS-92-174). Carnegie Mellon University, Pittsburgh, USA, 1992.