

PROTOS - A Rational Reconstruction

Peter Clark
The Turing Institute
36 N.Hanover St
Glasgow, UK

1987

1 Introduction

Recently, exemplar-based representations have been advocated as a useful alternative to the more traditional concept representation of a general set of conditions combined into conjuncts and disjuncts, as commonly used in expert systems (eg. [8]) and machine learning research (eg. [11]). Such exemplar-based schemes overcome several deficiencies of the traditional representation scheme, but raise their own issues.

One contemporary system utilizing an exemplar-based scheme is PROTOS ([12],[2]). PROTOS is unique in that exemplar selection and matching is based primarily on techniques using domain knowledge and inference, rather than statistical methods. As a consequence, besides illustrating a knowledge-based approach to reasoning with exemplars, its base of domain knowledge provides comprehensible explanations for its decisions. In addition to using exemplars, PROTOS is designed as a learning apprentice acquiring its domain knowledge through interactive problem-solving.

This paper is based on a simplified reconstruction of PROTOS, and is intended to be partly descriptive and partly discursive. In the first part, the motivation for using exemplar-based representations is presented. In the second part a description of PROTOS is given, and finally a discussion on various aspects of the system presented.

Two prefacing comments should be made. Firstly concerning jargon, literature on PROTOS refers to examples as ‘instances’ or ‘cases’, the values of their attributes as ‘features’, and the class to which they belong as their ‘category’. These words (example/instance/case, attribute-value/feature, class/category) are considered synonymous in this document. Secondly, any reconstruction from published papers involves a strong degree of interpretation to achieve a working, consistent system - thus the reconstruction, besides being very simple, undoubtedly deviates from the original in many other respects also. Comments about ‘PROTOS’ refer to the reconstructed version unless otherwise stated.

2 Exemplar-based schemes

2.1 Motivation

Representing concepts by a set of general conditions (combined into conjuncts and disjuncts) has already proved useful for certain types of problem-solving (eg. [11], [8]).

However, such a representation has also caused difficulties. This is particularly highlighted by the knowledge acquisition bottleneck encountered in expert system construction, where experts frequently find it difficult to express their knowledge in this form. One approach to easing this bottleneck has been to apply inductive learning techniques to acquire such concept descriptions from examples (eg. [9],[13]). However, although this concept representation is amenable to learning from examples (due to the ease of defining an appropriate search space and operators), it still presents difficulties in both fully capturing the expert's understanding of a concept and in enabling him or her to verify rules produced in this way.

As an alternative, exemplar-based approaches have been proposed. These approaches are based on the theory that most natural concepts are not well-defined but rather are based on relationships that are typical or only on the average true. As a consequence, a concept model is defined by a set of *prototypical examples*, rather than a general statement.

There are several reasons for preferring this approach :

1. Although the usual AI concept representation can capture concepts in certain artificial domains (eg. 'prime' in number theory, [7]), it has difficulty capturing more natural concepts due to their inherent polymorphy. Bareiss, Porter and Wier cite example concepts of 'friend', 'science' and 'pollutant' - it is difficult to imagine how they can be usefully represented by a general statement of necessary conditions.
2. Problem-solving systems, especially when coupled with inductive learning tools, have concentrated on the task of classification of new cases. A good representation of a concept should also support other functions, including example generation, explanation of the relevance of different features, and assessing degree of membership of a new example. Such functions are poorly supported when representing concepts by a general statement. Exemplar representations which may be able to perform these tasks efficiently.
3. Exemplar-based representations, due to their basis on specific instances, are able to focus interaction with an expert in terms of examples rather than generalities. This form of interaction is generally preferred by experts.
4. There is considerable psychological support for exemplar-based learning and problem-solving (see [1] for a review).

In the following, we consider the performance task of the system to be *classification*. In this case, the problem for exemplar systems is to select the exemplar best matching a new case thereby predicting the new case is of the same class. It should be noted that, having selected a best matching exemplar from memory, the performance task could be more complex - for example if the exemplars were previous plans then the performance task might be to modify the retrieved old plan to suit the current problem. These types of performance tasks, when using exemplar-like schemes, require substantial analogical reasoning following retrieval which we do not consider in this document (see for example [3], [5], [4]).

2.2 Classical representations

To contrast the use of classical and exemplar-based concept models, consider what such models might look like for the categories 'bird' and 'elephant' given the following training

data :

attributes				class
size	legs	colour	habitat	
big	talons	brown	moorland	bird
small	big-claws	white	sea	bird
v.small	small-claws	grey-white	sea	bird
small	big-claws	cream	sea	bird
v.big	short	grey	swamps	elephant
v.big	short	darkgrey	plains	elephant

A classical concept representation based on this data might look :

if sea then bird
if talons then bird
if v.big then elephant

(These rules might be generated using a rule induction algorithm, such as the AQ algorithm [10]). Although these rules can classify new examples, they have certain deficiencies:

- There is no notion of degree of membership of an example with a class
- Information about other features to expect of birds and elephants has been thrown away - thus these rules cannot be used to generate new examples.
- Correlations between features are not represented, eg. within the category birds, 'small' & 'sea' co-occur, and there are no sea-birds with talons.
- The justification for the rules is weak, based purely on statistical measures.

2.3 Probabilistic representations

An alternative representation of a concept, avoiding some of these pitfalls, is to use a probabilistic representation where each feature has associated weights attached to it for each possible class. Given a new example, the weight for each class in turn is computed by summing the individual contributions from each attribute of the new case.

$$\text{Weight Class}_i = \sum_j W_{ij} A_j$$

where W_{ij} is the weight that attribute j should contribute towards predicting class i . The class with the highest total is the predicted class. This method is a form of linear discriminant analysis, and has been a useful technique in pattern recognition. However, such methods do not solve all the above problems, for example information about feature correlations within a single class remain lost¹.

¹Consider 4 training exemplars, 2 of class A with features {a b} and {c d} respectively, and 2 of class B with features {a c} and {b d}. A new case with features {a b} would be considered equally likely of class A or B under linear discriminant analysis, but of class A (the correct class) if an exemplar approach was used.

2.4 Exemplar representations

A third possibility is to use exemplar representations. In these schemes, a concept is represented by one or more prototypical examples. Classification of new examples is performed by finding the best matching prototype and predicting the new example will be of the same class. There are two main issues for these systems:

- Which instances should be taken as exemplars?
- What is meant by a ‘good match’ of a new instance and exemplar?

The most primitive technique is to simply store all the training instances as exemplars. A good match is defined as being when the class of the matched exemplar is also that of the new case. How much weight should we put on each featural match? One method is simply to spread weight uniformly (ie. the best overall match maximizes the number of featural matches). However this can be improved on by apportioning more weight to features which are strongly correlated to the exemplar’s class, and less weight for those which seem uncorrelated. These ‘correlation weights’ can be calculated using linear discriminant methods similar to those used in probabilistic representations.

This crude approach is both memory intensive and slow, and it is difficult for an expert to comprehend a single concept represented by hundreds of exemplars. Kibler and Aha ([6]) illustrate how the number of exemplars can be greatly reduced without seriously damaging classificational accuracy using two alternative algorithms :

1. **Growth (additive) algorithm:** As new instances are processed, store (incrementally) only those training instances that were *not* correctly classified.
2. **Shrink (subtractive) algorithm:** Firstly, consider all instances as exemplars. Then each instance, in turn, is tested to see if it would be correctly classified by the remaining instances excluding itself. Those that would be correctly classified are removed.

As a consequence, they found a training set of 220 examples could be adequately represented by tens of exemplars in their experimental domain, despite the simplicity of these algorithms.

However, these schemes still suffer from several deficiencies :

1. The weight contribution of each feature match towards the overall instance-exemplar match is calculated statistically, measuring the feature’s correlation with the exemplar’s class. This calculation is inaccurate with small numbers of training examples, and still only approximate as the number increases.
2. Some features are not syntactically identical, yet should still be considered to match, for example
 - ‘big’ should match with ‘large’
 - ‘dark grey’ should be considered to partially match with ‘grey’
 - ‘metal’ should match with ‘wood’ if the significant property is that they are rigid materials

Addressing these two issues has been a central focus in PROTOS. The solution offered is, at the most general level of description, to include a base of domain knowledge to reason with and reason about. We now proceed to describe PROTOS in more detail.

3 PROTOS

3.1 Overview

A schematic diagram of the reconstruction of PROTOS is shown in Figure 1. The flow of control can be summarized as follows :

1. Get a new case from the user
2. Find the exemplar which best matches it
3. Present the category of this exemplar as the predicted category of the new case
4. Ask user if the prediction is correct. If not, goto 2 and exclude exemplars of this failed category. Else continue.
5. Ask the user to explain features of the new case which don't match with the exemplar, and why they might occur in instances of the category.
6. If the exemplar and new case are sufficiently similar then merge them, else make the new case into a new exemplar in the database.
7. Stop

In the following we describe the central tasks in PROTOS, and how they are approached in a knowledge-based fashion. These tasks are :

- Matching a new case with an exemplar
- Reaction to a incorrect prediction
- Deciding when a new case should merge with an exemplar, and when it should form a new exemplar
- Acquiring the domain knowledge

3.2 Matching with exemplars

As mentioned earlier, two deficiencies with the matching-by-summing-weights method were :

- The poor validity of weights calculated using statistical methods when based on few examples
- The inability to detect indirect (non-syntactic) matches between features (eg. 'dark-grey' and 'grey').

PROTOS addresses this first problem by calculating initial weights using an *analysis of the justifications* that the features should be present in members of a category. The second problem is addressed by making two passes at matching. The first pass performs a standard weight summing approach to find best matching exemplars, then a second pass is performed which attempts to improve these matches by finding *indirect* correspondances between features using its domain knowledge. These two processes of selecting reminding weights and knowledge-based matching are now described.

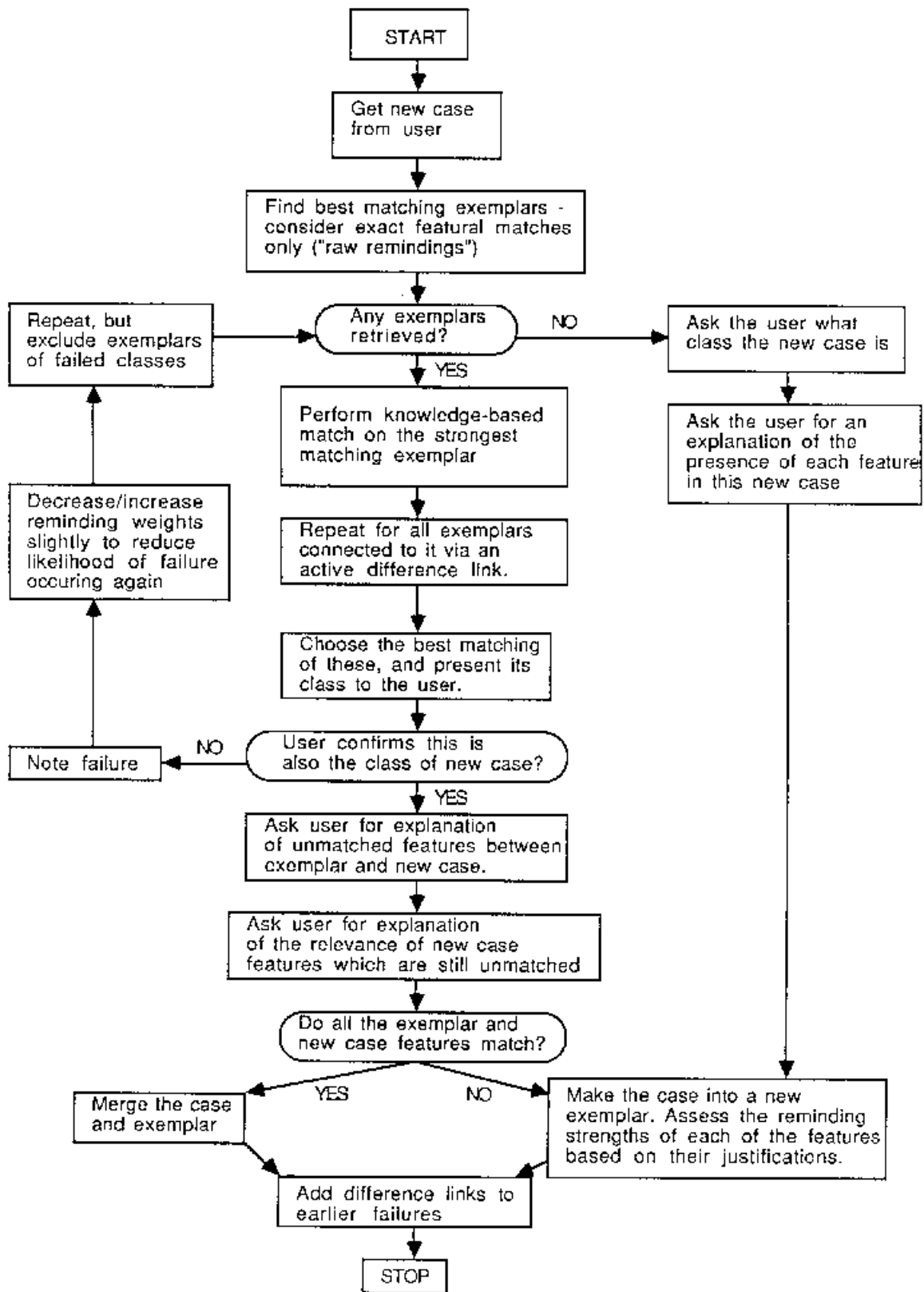


Figure 1: The Reconstructed PROTOS algorithm in detail.

3.2.1 Reminding weights

Following the terminology in PROTOS, we refer to the weighted contribution which a featural match makes towards an overall newcase-exemplar match as the *reminding weight*. Rather than calculate reminding weights statistically, PROTOS first asks the expert to explain *why* the presence of a feature should be associated with an exemplar's category, then assesses the strength of the explanation. A strong explanation implies the reminding weight should be high, a weak explanation implies it should be low.

The strength of the explanation is assessed according to several criteria. These are:

- The number of inferences in the explanation
- The types of inferences. In PROTOS, inferences are categorized into types (the 1986 version of PROTOS had 6 types: definition, enable, implies, specialization, causes, part-of), and some types (eg. definition) are considered to be more reliable than others (eg. implies).
- Whether there are any other alternative explanations
- Whether there are any competing classifications, ie. the feature can also be shown by a chain of explanation to imply a different category.

Unfortunately, the two published papers on PROTOS do not give the numerical parameters required for this calculation, nor the combining functions which were used. The parameters used in reconstruction are given in section 4.

3.2.2 Knowledge-based pattern matching

When given a new case, PROTOS first selects candidate good matches by the traditional method of summing weights of features matching exemplars for each exemplar, then selecting the exemplar(s) with highest total.

Following this, PROTOS then makes a second pass at matching to try and improve the match using domain knowledge. PROTOS looks for features which don't match directly but can be shown to be 'equivalent', hence match indirectly. Two features are considered 'equivalent' if they can be shown (using inference rules) to imply a common feature, eg. if $a \Rightarrow b \Rightarrow c$, and $d \Rightarrow c$, then a and d are equivalent. (NB. c will be a common generalization of a and d in the special case that the ' \Rightarrow ' rules are 'isa' relations, but not in the general case)².

For example, PROTOS will consider 'grey' to match with 'dark grey' if it knows the inference rule 'dark grey isa specialization of grey'. In addition, because the match is indirect, its reminding strength is lowered from that which a direct match would produce. This reduction is a function of the strength of the explanation showing the features are equivalent, presumably calculated in a similar way to that for reminding weights.

Unfortunately, allowing inferential as well as syntactic matches raises two problems. Firstly, with a large enough database of rules, almost any feature can be shown to relate in some way (possibly very indirectly) to some other feature. Consequently, without constraining the search in some way, 'ghost' matches can be found which are not really justifiable. Secondly, computational resources are not sufficient anyway to allow a

²The literature on PROTOS is not entirely clear on what 'equivalent' means; this definition is the best that can be concluded from examples in the papers.

complete search of possible matches, again requiring the search be constrained. This is achieved by limiting the resources to expend on knowledge-based matching to an upper bound. This bound is a function of (amongst other things) the degree of direct match between the case and the exemplar.

3.3 Reaction to failure

Having selected a best-matching exemplar, PROTOS predicts that the new case will also be in the same category as that of the exemplar. However, despite the use of explanations to assess reminding weights, and the use of knowledge-based pattern matching for exemplar selection, PROTOS is sometimes incorrect in its prediction. In order to avoid making the same mistake again, two procedures are used :

1. **Altering reminding weights:** In the event of recalling an exemplar of the incorrect class, the reminding weights on all the exemplar's features which contributed towards the reminding (ie. matched the new case) are decreased. Similarly, the exemplar's features *not* matching the new case are considered to be more important than originally thought, and have their weights slightly increased.

In this way, PROTOS is attempting to use a **negative feedback loop** to end up with an optimal set of reminding weights connecting features and exemplars.

2. **Adding difference links:** After failure, PROTOS retries again for a match (avoiding exemplars of the failed category this time). When eventually the correct match is found, a 'difference link' is added connecting the correct and failed exemplars together. This link is labelled with the featural differences between the two exemplars, and can only be traversed when a new case has all these differences as features.

During future problems, if the failed exemplar is ever selected again as a potential match, PROTOS will also examine all other exemplars connected to it by active difference links.

Thus, difference links serve as a kind of 'patch' or 'exception episode', noting where matches were overlooked in the past so that they will not be overlooked in the future.

3.4 Merging a new case and an exemplar

An exemplar in PROTOS is a set of features plus the justification for their occurrence in the exemplar's category. When considering whether to merge a new case with an exemplar or form a new exemplar from it, PROTOS examines the justifications. If *all* the justifications associated with an exemplar also apply to the new case, then the new case and exemplar are merged - this occurs if all the exemplar's features match (directly or indirectly) with the new case's features. Otherwise, the new case becomes a new exemplar.

During merging, features common to both the exemplar and new case are retained. Those features which matched indirectly are replaced by a new feature implied by both and used in the justification of the exemplar's category membership. For example, if the exemplar feature $f \Rightarrow g \Rightarrow h \Rightarrow \text{category}$, and the new case feature n matches with f by the relations $n \Rightarrow g \Leftarrow f$, then f and n are replaced by g during the merge. Note that this

operation is one of generalization in the special case that the inferences ‘ \Rightarrow ’ are all ‘isa’ relations, but not in the general case.

3.5 The learning apprentice

PROTOS relies heavily on domain knowledge for its operation. This is acquired from the expert during interactive classification of new cases and illustrates one of the biggest advantages of exemplar-based representations, namely that dialogue can be focussed on specific cases rather than at the level of generalities. Such focussed dialogue is considered preferable to an expert.

Inference rules are acquired at two main points during processing :

1. After selecting a best matching exemplar, PROTOS asks the user about the unmatched features in the exemplar and new case. The user can supply rules at this point to enable PROTOS to show their equivalence. Here the dialogue is about feature-feature relationships.
2. If features remain unmatched, PROTOS asks the user for an explanation for why each of them should be expected to occur in instances of the new case’s category. Here, the dialogue is about feature-category relationships.

PROTOS interacts with the user in other ways also, which will not be documented here.

3.6 Prototypicality

For the sake of completeness, a feature of the original PROTOS not implemented in the reconstruction should be mentioned. This was that, in the original PROTOS, features common to all exemplars of a category were linked to a node representing the category, rather than to all the exemplars within the category. As a consequence, categories as well as exemplars could be recalled.

There was a threshold reminding strength below which a reminding (ie. a match) was deemed not worth considering. Given a new case, the best matching exemplars were searched for as described earlier. In the event of the matching strength of the best matching exemplar being below this threshold, then the exemplar was *not* selected. Instead, the best matching *category* node was located, and then the most prototypical exemplar within the category selected. The prototypicality of an exemplar is determined by the extent to which its features overlap those of other category members.

4 Discussion

The main features of PROTOS have been described. In this section, some discussion on PROTOS is given.

4.1 Functions and parameters

One of the least satisfying features of PROTOS is the large number of parameters and functions it relies on to operate. Some of these are listed below, along with the values taken in the reconstruction. None of these functions and parameter values are documented in the papers on the original system.

4.1.1 About reminders

1. What strength of justification is attached to each different type of inference (\Rightarrow)?
In reconstruction, they all were given the same strength of 1 point.
2. How is the strength of a explanation relating an exemplar's feature to its category calculated?
In reconstruction, strength = $10 - L$, where L is the number of inferences made in the explanation.
3. How is the initial reminding weight calculated from an examination of this explanation strength?
In reconstruction, initial reminding weight = $10 \times$ explanation strength.
4. In the case of user rejecting PROTOS's diagnosis (ie. the best matching exemplar NOT being in the same class as new case), the reminding weights on features linked to the exemplar are changed (decrease weights on features present in the new case, increase weights on those not). How large are these weight changes?
In reconstruction, increase/decrease by 5 points.

4.1.2 About knowledge-based matching

1. How is the strength of a match between a new case feature and an exemplar feature calculated?
In reconstruction, get 10 points per direct match and $(10-L)$ points per indirect match where L is the number of inferences required for this indirect match. eg. if a and b match via inferences $a \Rightarrow x \Rightarrow y$, and $b \Rightarrow y$, then match strength = $10-3 = 7$.
2. How are the strengths of individual feature matches combined to assess the overall similarity between a new case and an exemplar (ie. the overall strength of a knowledge-based match)?
In reconstruction, simply sum the contributions together.
3. What is the heuristic function which limits the effort expended in knowledge-based pattern matching?
In reconstruction, matching was exhaustive as the knowledge base and number of cases considered was always small.

These were the functions and parameters required for the simple reconstruction. It should be noted that several more have not been mentioned here which are necessary for the full PROTOS system.

This large number of parameters raises several questions. Firstly, can they be justified in any theoretical sense? For the purposes of reconstruction they were chosen by trial and error until the desired behaviour was achieved. The danger is that, if they remain as ad hoc functions, the inadequacies they seek to overcome may actually be made worse. For example, one motivation for calculating weights using a knowledge-based analysis of proofs rather than a statistical method is that poor statistical approximations will be avoided. However, it may be the case that as many, if not more, approximations

have been introduced by replacing statistical methods with an ad hoc knowledge-based analysis.

Secondly, there is the more analytical issue of whether these functions do in fact produce the behaviour that is intended. This will be discussed in the following section in more detail.

Thirdly, there is the practical issue that a system relying on complex functions and methods is difficult to both analyse and reconstruct - this raises questions about its value as a useful learning technique, even if it is demonstrated to be successful in experiments.

4.2 Learning, interdependence and feedback

PROTOS is designed to be an adaptive system, making use of a *negative feedback loop* to improve performance. If PROTOS incorrectly predicts a new case's category, it will increase/decrease reminding weights on the best matching exemplar's features to make it less likely that this failure will reoccur. Ideally, as the number N of cases seen by the system grows, failures should become less and less common and disappear completely as N approaches infinity. If this were to happen, then the weights on exemplars could be said to have converged on optimal values.

However, there are particular problems for exemplar-based systems which makes such an ideal difficult to achieve. Primarily, *the components of the system interact highly and are strongly coupled together*. Consider PROTOS deciding a new case matches exemplar E best; this statement is not just dependent on the new case and the exemplar E , but it is also dependent on all the *other* exemplars in the database, their current reminding weights, and the inference rules available. PROTOS had to involve all these to decide that the new case matched E best. If we were to add a new inference rule, a new exemplar or change a reminding weight somewhere in the system, the statement "new case best matches E " may become false.

Of course, most systems are partially subject to this interdependence - however this interdependence is at a much simpler, more predictable level with classical concept representations. Consider adding a new classification rule to a set of traditional classification rules - it is immediately obvious which other rules the new rule might conflict with, and a simple decision procedure can be used to choose between conflicting predictions should they arise. The problem with exemplar-based systems, and particularly PROTOS, is that the effects of (say) adding a new inference rule are almost impossible to predict as such changes can affect future classifications in many indirect ways.

Why is this unpredictability a problem? Well, consider for a moment a simpler adaptive system using negative feedback (eg. an electronic circuit). To ensure negative feedback achieve optimality, two considerations need to be made :

- The amount of feedback needs to be carefully controlled to ensure convergence rather than oscillatory behaviour
- Any more structural change in the system may mean that near-optimal parameters are no longer near-optimal, and consequently need to be updated

The second point here is the most important. During operation, PROTOS is constantly making changes to its knowledge base which will affect the behaviour of the entire system. If not attempt is made to preserve old behaviours, then negative feedback loses its effect; negative feedback will not work for systems which keep altering their structure,

as the conditions which they are trying to optimize under keep on changing (ie. the notion of ‘best’ keeps on changing). This can be overcome by appropriately changing parameters along with structural changes such that behaviour is preserved, but PROTOS makes no attempt to do this - indeed, it would be virtually impossible to do so, as the interactions between parts of the system are highly complex.

As a consequence, attempts to use negative feedback in PROTOS are problematic, because the conditions under which PROTOS tries to optimize are continually shifting. It is difficult to see how these difficulties can be avoided, as the components in exemplar-based systems are highly coupled and interdependent, consequently their interaction is difficult to predict. It is quite frequently the case that weight changes made for the n th new case will undo some of the changes made for the $(n-1)$ th new case, or that changes in the knowledge-base will cause previously successful classifications to fail in future.

4.3 The learning apprentice

Finally, some words about problems encountered with the learning apprentice aspect of PROTOS should be mentioned.

The first problem encountered is that user frequently ‘lies’ to the system. This isn’t out of any malicious intent to confuse, but rather because people often state rules which are only valid in the context in which they were stated. For example, the user may say “glue holds together wood” as an part of the explanation for the presence of glue in a model aircraft kit - however this rule is not universally true (eg. if the glue is already dry before use, the wood very heavy/absorbent/wet before use, etc). Or the user may say “feet enable walking” in the context of people, enabling PROTOS to conclude that tables can also walk. If user-supplied rules are used in a more general context than that in which they were supplied, then they may be applied incorrectly. (It isn’t stated in PROTOS literature whether rules are made globally available or confined to the context in which they were presented).

The second problem is that of words having multiple meanings. Synonym problems (eg. ‘big’ and ‘large’ meaning the same thing) can be avoided by using a synonym dictionary - a greater problem is using the *same* word for *different* meanings. The user may use the word “file” for paper files, computer files, nail files and metal files, causing inference rules to be applied incorrectly. Often the meaning of a word is relative (eg. ‘big’). The user may use the same word to refer to an instance (“fred isa elephant”) or a set (“elephant isa species”). These problems of dialogue are well known in work on natural language processing - they also need to be addressed in a learning apprentice system.

Thirdly, it often turns out to be quite difficult to give rules to PROTOS to explain features. Consider, for example, trying to justify why houses have windows or roofs. Saying “windows let the people look out” isn’t a good enough explanation for PROTOS, as this explanation mentions nothing about houses. Apart from simply saying they are associated together (“windows partof house”), a justification might start:

“windows enable looking_out”

“people like looking_out”

“houses contain people” Now what? The user still needs to explain that houses contain things enabling people to do things they like (eg. looking out). Even if the user does express this in some rule notation, they still haven’t really said what they mean as the statement implies houses also contain TV, swimming pools, palm beaches plus any other

heart's desire. Of course, if the user thinks hard enough he or she can usually express fairly much what they mean - the point is that a simple dialogue over explanations can still be quite a knowledge acquisition bottleneck.

A final problem is that in order to explain things, the user needs to have a good idea of what PROTOS already knows so that he/she doesn't have to recite the entire explanation. With person-person communication, this is usually no problem as people have a fairly good idea of what other people already know/don't know. However, with person-PROTOS communication, this is not the case and is a barrier to enabling explanation. As a user of PROTOS, I need to know what PROTOS already knows in order to see how to provide a useful explanation. In the reconstruction this is achieved by PROTOS simply printing out the knowledge-base before asking the user for an explanation, but this solution is not very satisfactory.

5 Conclusion

PROTOS is an ambitious contemporary system attempting to reason with exemplars in a knowledge-based rather than statistical fashion. As a consequence, it can produce comprehensible explanations for its decisions and carry out a highly focussed dialogue with the user based on specific cases. This makes it additionally useful as a knowledge acquisition tool as well as a classifier. Unlike many traditional learning systems it is incremental and acquires a structured model of the concepts it is being taught over time.

Unfortunately, PROTOS relies on a relatively large number of ad hoc parameters and functions to perform, making it difficult to show by analysis that it will in fact achieve its aims. Problems with negative feedback occur as components of the system are highly coupled, resulting in continuously changing conditions under which optimization is to occur. Finally, although the bandwidth of communication with the user has been increased from simply presentation of examples, this opens up additional problems - particularly caused by the requirement to understand the context of the dialogue in order to correctly interpret the user's statements.

References

- [1] E. R. Bareiss and B. W. Porter. *A survey of psychological models of concept representation*. Technical Report AI86-50, The University of Texas at Austin, Computer Sciences Department, Austin, TX, 1987.
- [2] E. R. Bareiss, B. W. Porter, and C. C. Wier. PROTOS: an exemplar-based learning apprentice. In P. Langley, editor, *Proc. 4th International Workshop on Machine Learning*, Kaufmann, Ca, 1987.
- [3] M. Burstein. Concept formation by incremental analogical reasoning and debugging. In J. G. Carbonell, R. S. Michalski, and T. M. Mitchell, editors, *Machine Learning, vol. 2*, Tioga, Palo Alto, Ca, 1986.
- [4] J. G. Carbonell. Derivational analogy: a theory of reconstructive problem solving and expertise acquisition. In J. G. Carbonell, R. S. Michalski, and T. M. Mitchell, editors, *Machine Learning, vol. 2*, Tioga, Palo Alto, Ca, 1986.

- [5] J. G. Carbonell. Learning by analogy : formulating and generalizing plans from past experience. In J. G. Carbonell, R. S. Michalski, and T. M. Mitchell, editors, *Machine Learning, vol. 1*, Tioga, Palo Alto, Ca, 1983.
- [6] D. Kibler and D. W. Aha. Learning representative exemplars of concepts: an initial case study. In P. Langley, editor, *Proc. 4th International Workshop on Machine Learning*, Kaufmann, Ca, 1987.
- [7] D. Lenat. The role of heuristics in learning by discovery: three case studies. In J. G. Carbonell, R. S. Michalski, and T. M. Mitchell, editors, *Machine Learning, vol. 1*, Tioga, Palo Alto, Ca, 1983.
- [8] J. McDermott. R1: a rule-based configurer of computer systems. *Artificial Intelligence*, 19(1):39–88, 1982.
- [9] R. S. Michalski and R. Chilausky. Learning by being told and learning from examples: an experimental comparison of the two methods of knowledge acquisition in the context of developing an expert system for soybean diagnosis. *Policy Analysis and Information Systems*, 4(2):125–160, 1980.
- [10] R. S. Michalski and J. Larson. *Incremental generation of VL_1 hypotheses: the underlying methodology and the description of program AQ11*. Technical Report ISG 83-5, The University of of Illinois at Urbana-Champaign, Department of Computer Science, Urbana, 1983.
- [11] T. M. Mitchell, P. Utgoff, and R. Banerji. Learning by experimentation: acquiring and refining problem-solving heuristics. In J. G. Carbonell, R. S. Michalski, and T. M. Mitchell, editors, *Machine Learning, vol. 1*, Tioga, Palo Alto, Ca, 1983.
- [12] B. W. Porter and R. E. Bareiss. PROTOS: an experiment in knowledge acquisition for heuristic classification tasks. In *Proceedings of IMAL 1986*, Université de Paris-Sud, Orsay, France, 1986.
- [13] J. R. Quinlan, P. J. Compton, K. A. Horn, and L. Lazarus. Inductive knowledge acquisition: a case study. In *Proceedings of the second Australian Conference on the Applications of Expert Systems*, pages 183–204, New South Wales Institute of Technology, Sydney, 1986.