Prepublication draft of Webb, G. I. (2002). **Integrating Machine Learning with Knowledge Acquisition**. In Leondes, C. T. (Ed.), *Expert Systems*, volume 3, pages 937-959. San Diego, CA: Academic Press.

Techniques for integrating machine learning with knowledge acquisition

Geoffrey I. Webb

School of Computing and Mathematics

Deakin University

Geelong, Vic, 3217,

Australia

Tel: +61 3 5227 2606

Fax: +61 3 5227 2028

Email: webb@deakin.edu.au

INTRODUCTION

Knowledge acquisition is frequently cited as the greatest bottleneck in the development of expert systems. Two primary approaches to knowledge acquisition are elicitation of knowledge from experts (traditional knowledge acquisition) and machine learning. Both approaches have strengths and weaknesses. Experts can draw upon practical experience and both domain specific and general knowledge. In addition, they often have access to well established procedures relating to the target domain. Elicitation of knowledge from experts is limited, however, by the difficulty of articulating knowledge; reluctance in some circumstances to share knowledge; preconceptions and biases that might inappropriately influence an expert; and the limits of an expert's knowledge. In contrast, machine learning systems provide a capacity to infer procedures from examples; can perform extensive logical analysis; and are not subject to the same types of preconceptions and biases as an expert. They are hampered, however, by limited access to general and domain-specific knowledge and the difficulties of obtaining comprehensive example sets. Further, machine learning is only possible where much of the knowledge acquisition task has already been completed. Machine learning requires a description of the problem domain and collection of example cases from which to learn. In attribute-value machine learning, the domain description consists of a set of attributes and their allowable values along with a collection of class values. These, together with the formalism used for expressing the decision procedures, specify a space of possible solutions that the system might 'learn'. The learning system then explores this space of solutions seeking one that best fits the training examples. If a suitable space of possible solutions is specified, learning is relatively straightforward. If a poor solution space is specified, effective learning is impossible. Thus, machine learning requires prior ontological analysis and the specification of a suitable class of models to explore.

It is clear that this pre-learning process is crucial. A domain expert will usually provide critical input into this process. This is not the end of the domain expert's involvement in knowledge acquisition by machine learning, however. In practical applications of machine learning it is often necessary for an expert to review the rules that a machine learning system creates (see, for example, Buntine & Stirling, 1991). These rules may require modification due to pragmatic or other considerations quite outside the formal factors that a machine learning system is able to consider. For example, social or ethical considerations may make unacceptable a policy to refuse loans to a specific class of loan applicant, no matter how compelling the evidence is that the particular class of applicant represents a poor risk. Review by the expert will also frequently lead to the identification of deficiencies in the domain specification, which will necessitate a cycle involving modification of the domain specification and re-application of machine learning, followed by another domain expert review. Most machine learning tools provide little support for these interactions with the expert.

In addition to assisting the expert in processes that must be carried out to support machine learning, there is also a need for facilities to enable the expert to directly assist the machine learning system during the induction process.

However, there are a number of obstacles to providing such support. The first of these is communication. Any sophisticated interaction between an expert and a machine learning system will require the two to communicate with one another. But, such communication requires mechanisms beyond those normally supplied by machine learning systems.

A second difficulty is that the machine learning system usually provides a very shallow knowledge-base. Without any means of obtaining explanations and justifications of the rules that are presented, an expert can find these shallow rules difficult to assimilate into his or her more sophisticated models of the domain.

Interactions are further limited by the restricted forms of input to which most machine learning systems are restricted. Most attribute-value systems allow only the description of the attributes and of a set of examples. There is no provision for background knowledge or any other suggestions or constraints with which to guide the learning process.

A final obstacle is that machine learning systems can make quite arbitrary choices in situations where there are alternative solutions that equally well fit the example cases. Such situations are quite frequent in practice. These arbitrary choices may be acceptable in such situations if no other information is available and maximisation of predictive accuracy is the primary consideration, as is usually considered the case in machine learning research. However, an expert will often be able to discriminate between the alternatives on other grounds. In this context, the systems' arbitrary choices may reduce the comprehensibility or acceptability of the inferred rules for the expert or may lead to less useful rules than if the expert's judgements are utilised.

A large number of systems and techniques have been designed to tackle aspects of interaction between a user and a machine learning system. Most of these are oriented toward interactive use by a sophisticated knowledge engineer (Attar Software, 1989; Davis & Lenat, 1982; De Raedt, 1992; Morik, Wrobel, Kietz & Emde, 1993; Nedellec & Causse, 1992; O'Neil & Pearson, 1987; Schmalhofer & Tschaitschian, 1995; Shapiro,

1987; Smith, Winston, Mitchell & Buchanan, 1985; Tecuci & Kodratoff, 1990; Wilkins, 1988).

This paper describes The Knowledge Factory, a computer-based environment that allows a domain expert to directly collaborate with a machine learning system at all stages of the knowledge acquisition process. We have endeavoured to take the above considerations into account in designing this system. It is distinguished from previous such systems by its orientation toward direct use by domain experts with minimal computing sophistication.

THE KNOWLEDGE REPRESENTATION SCHEME

Sophisticated knowledge representation schemes, such as first-order logic, are not appropriate for use by non-knowledge-engineers (see, for example, Kodratoff & Vrain, 1993). Extensive training and experience is required to master such formalisms. Instead, The Knowledge Factory uses simple attribute-value case descriptions and production rules. Example cases are described by simple vectors of attribute values. Rule antecedents (conditions) are conjunctions of simple tests on attributes. The consequents (conclusions) are simple classification statements.

Fig 1 about here

Examples of rules are provided in Fig. 1. These examples illustrate rules learned from data on the diagnosis of renal disease used in research with Geelong Hospital (Agar & Webb, 1992). Attributes are either categorical or ordinal. For categorical attributes, set membership tests are used (although set notation is avoided). All tests in the first rule in Fig. 1 relate to categorical attributes. For ordinal attributes, tests specify a range of allowable values. The attributes *age* and *prodocytic_fusion_ef* tested in the second rule of Fig. 1 are both ordinal. The condition part of a rule (the tests between the keywords **IF** and **THEN**) is satisfied for a case if all of the individual tests are satisfied. If the condition is satisfied then the conclusion for the rule is taken to apply to that case.

One complication that must be handled by a machine learning system is missing values in the data. The Knowledge Factory recognises two types of missing value, *unknown* and *unobtainable*. The latter is treated as a distinct value, that may be tested in a rule, *unobtainable* having the same status as a normal value. The first test in the third rule in Fig. 1 allows unobtainable values for the attribute *age*. Any case with an unobtainable value for *age* will fail the condition for the second rule, which does not explicitly allow unobtainable values for this attribute. Unobtainable values should be used where the absence of a value needs to be considered during the decision process, for example, because different attributes should be considered if an attribute normally used in the decision making process does not have a value. In contrast, unknown values pass any test on an attribute. In consequence, the test for the second rule in Fig. 1 will succeed for a case with an unknown value for *age*, so long as all of the tests on other attributes succeed. The condition on *hepatitus_B_antibody* in the second rule of Fig. 1 will only succeed for a case with an unknown value.

A summary line follows the consequent of each rule. This displays the number of training examples that the rule correctly classifies (*Positive*), the number incorrectly classified (*Negative*) and the value currently assigned to the rule by the system (used, under some settings, when resolving between multiple rules that cover a case).

In the current version of the system, the rule sets are flat. All rules consequents relate to the same attribute, called the decision attribute. This attribute cannot be used in the

antecedent of a rule. While it might be appropriate to slightly relax these constraints, it would not be appropriate to allow complex chaining to any great depth, as non-knowledge-engineers are unlikely to readily master the complexities of such a rule base. Attempts to incorporate more sophisticated knowledge representation devices, such as model construction operators (Morik, Wrobel, Kietz, & Emde, 1993), are also viewed as likely to impose an undesirable barrier to use for those without extensive training in knowledge engineering.

We have conducted informal experimentation with two sets of experts—medical practitioners and financial analysts. The former have had little computing expertise and the latter have had substantial computing expertise but no background in knowledge-engineering. These experiments show that these experts adapt readily to this form of knowledge representation. They have no difficulty in interpreting existing rules, or specifying new rules or examples.

MACHINE LEARNING TECHNIQUES

The Knowledge Factory uses the DLGref learning algorithm (Webb, 1993). This is an attribute-value classification learning algorithm that supports refinement of existing rules. Attribute-value classification learning algorithms take as input a *training set* of pre-classified examples. Each of these examples is described by a vector of attribute-values and is labelled with the correct class. The algorithm forms a model that relates combinations of attribute-values to classes. Note that while this abstract description of machine learning describes the models that it produces as classifiers, classification can be used to model many forms of discrete decision making. Thus, classification learning encompasses many useful learning tasks.

Generalisation, Specialisation, and Cover

The Knowledge Factory makes extensive use of two operations on rules: generalisation and specialisation. The following provides a brief introduction to these operations.

The classification rules used by The Knowledge Factory have the form IF *condition* THEN *classification*. If the condition of a rule is satisfied by a case then the rule is said to *cover* that case.

A rule *x* is a *generalisation* of a rule *y* if *x* necessarily covers all cases that *y* covers. For example, the rule *IF age>20 THEN old* is a generalisation of *IF age>30 THEN old*. A case covered by the second rule must also be covered by the first rule. To create a generalisation of a rule in the language used by The Knowledge Factory it is necessary to either make a constraint on an attribute less strict, for example by lowering the minimum value allowed as in the example above, or remove a constraint on an attribute, as with the generalisation from *IF age>20 and hair is fair THEN x* to *IF age>20 THEN x*.

If rule *x* is a generalisation of rule *y*, then *y* is a *specialisation* of *x*.

Note that by these definitions, a rule is both a generalisation and a specialisation of itself. A rule *x* is a *proper generalisation* of a rule *y* if *x* is a generalisation of *y* and *x* is not a specialisation of *y*. If *x* is a proper generalisation of *y* then *y* is a *proper specialisation* of *x*.

A rule x is a *least generalisation* (Plotkin, 1970) with respect to rule y and case z, if x is a generalisation of y and x covers z, and there is no rule r that is a proper specialisation of x, a generalisation of y and covers z. For example, *IF age>20 THEN old* is a least generalisation of *IF age>30 THEN old* with respect to a case with age 20.

A rule x is a *least specialisation* with respect to rule y and case z, if x is a specialisation of y and x does not cover z, and there is no rule r that is a proper generalisation of x, a specialisation of y and does not cover z. For example, *IF age>30 THEN old* is a least generalisation of *IF age>20 THEN old* with respect to a case with age 29 (assuming that only integer values are allowed for age).

DLGref

DLGref is a variant of the AQ (Michalski, 1993) learning algorithms. These algorithms form a set of classification rules incrementally. For each class a rule is sought that performs best on the remaining examples. Best performance is judged in a variety of ways, but usually involves covering as many examples of the class and as few examples not of the class, as possible. DLGref allows the user to specify a metric of good performance. Within The Knowledge Factory, two metrics are supported. The *max-consistent* metric does not accept any rules that cover examples of another class and prefers of the remaining rules those which cover the most examples of the positive class. *Laplace* is based on the Laplacian law of succession. It allows a trade-off between covering greater numbers of examples and covering small numbers of counter-examples. Using this metric, the value of a rule equals $\frac{p+1}{n+2}$, where p is the number of positive examples covered by the rule and n is the total number of examples covered by the rule. Having selected the best rule, all examples that it correctly classifies are removed from the training set and the process is repeated. The rules so selected are pooled together to form a set of classification rules.

The DLGref algorithm is presented in Appendix A.

Whereas most AQ-like covering algorithms search for each rule starting from very general rules and considering successive specialisations thereof, DLGref starts from highly specific rules and explores successive generalisations. The successive generalisations are generated using least generalisation. This process readily supports ordinal and continuous valued attributes, forms that cause difficulties for the specialisation-based search of traditional AQ algorithms.

Information Utilised by the Machine Learning System

An attribute-value classification learning system utilises two major types of information. The first is the set of attributes and their possible values. This, along with the formalism for expressing classifiers that the system employs, defines the set of possible classifiers that the system can form. Clearly, the system cannot form classifiers that utilise attributes about which the learning system has not been informed. The second type of information that is used is the training set. The learning system searches the space of possible classifiers that it is capable of forming, seeking a classifier that best classifies the examples in the training set. Clearly, the system is greatly influenced by the quality of the training set with which it is provided. If there are intricacies to classification in a domain that are not represented in the training set then the learning system cannot learn them.

TECHNIQUES

One of the primary design considerations for The Knowledge Factory was to allow direct interaction between the knowledge acquisition system and an expert with minimal computing expertise. This has been achieved by a number of previous knowledge acquisition systems, notably repertory grid based tools such as ETS (Boose, 1986) and the ripple-down-rules tools of Compton, Edwards, Srinivasan, Malor, Preston, Kang, & Lazarus (1992). However, a significant difference between The Knowledge Factory and these previous tools is that The Knowledge Factory's incorporation of machine learning frees it from a reliance on the expert having all encompassing knowledge of the domain. Previous systems have relied upon the expert being able to specify a suitable solution for any problem that the system may present. The expert, then, is viewed as all-knowing. All that is required is to help him or her to develop a formal model of the domain that captures the relevant aspects of his or her expertise. With the inclusion of machine learning, this view can change. Now the expert is viewed as having valuable insight into the domain, but this insight need not be all encompassing. Where the expert is unable to provide suitable models, the machine learning system can supply models instead. But the interactions may be more sophisticated than each simply producing part of a model, the two parts then being assembled.

An alternative scenario is that an expert may have expertise in some aspect of a domain but may not be able to express formal models that encode that expertise. That is, the expert may be able to solve a particular class of problems, but may not be able to specify a set of expert system rules to reproduce those solutions. In this case, the machine learning system can be used to produce a 'first draft' of a set of rules. The expert may then be able to critique and refine those rules. Thus, machine learning can be used to 'kick-start' the model specification and refinement process.

A further scenario is that even after the use of machine learning to produce a first draft, the expert may still be unable to specify appropriate formal models. In this case, he or she may be able to critique the machine learning system's models, enabling it to further refine its own models.

Finally, the expert may be able to specify some formal models initially, but these models may be incomplete or imprecise. The machine learning system may then refine this initial knowledge-base. The expert can review and revise the machine learning system's contributions in turn, starting an ongoing sequential process of refinement.

The commonalities of all these scenarios are an underlying view of the expert and the machine learning system as partners in the knowledge acquisition task, each of which has differing insights into a domain. The expert's insights are derived from training, experience and both general and commonsense knowledge. The machine learning system's insights are derived from analysis of a set of training examples.

Communication Mechanisms

Such pooling of insight clearly requires communication between the partners of a form not normally supported by machine learning systems. The Knowledge Factory uses example cases as the primary basis for communication. This case-based communication paradigm suits both partners in the collaboration. Experts in many domains are familiar with the use of examples, both to have points illustrated and for

illustrating points. For a machine learning system, example cases are the major source of evidence that is considered. Many of the decisions that a machine learning system makes can only be explained in terms of how a model relates to the provided training examples. With The Knowledge Factory's case-based communication, each partner uses example cases to communicate with the other.

Example windows

The primary mechanism used by the machine learning system is to provide windows that list the example cases that stand in a particular relationship to a rule or set of rules. Eight of these example windows are maintained. While these windows could be composed to answer specific queries from the user, or even to anticipate queries that the user might wish to pose, the complexity of creating such queries, and the difficulties of understanding system generated summaries of the queries that an ad hoc window might address, mitigate against the use of such mechanisms with our target users. Rather, windows that address the issues that are most likely to concern the users are maintained at all times. This constant display allows a user to familiarise himself with the meaning of each window, not having to devote attention to comprehending the meaning of a window each time that it is considered, as would be the case if windows presenting different types of information were generated dynamically by the system. The user is then able to attend to individual windows at his own discretion, ignoring those that do not bear upon his current concerns and utilising those that do.

The first example window simply lists all available examples. It is perhaps stretching the scope of the technique to include this window within the case-based communication mechanism, as, in contrast to the other example windows, it does not really serve a communication purpose. Rather, it provides the user with ready access to the available example cases.

The Indistinguishable Examples Window contains all examples for which there is an example from another class such that the system is not able to form a rule that distinguishes the two. This will occur if there is no attribute for which the two examples in question have different values. Unknown values do not count as distinct for this purpose. The presence of examples in this window indicates that it is not possible to form rules that correctly classify all examples given the current set of attributes. This indicates to the user that it might be desirable to consider revision of the current set of attributes.

Whenever a rule from the current knowledge base is selected, the Positive Examples Window lists all examples that the rule correctly classifies. The Counter Examples Window displays all examples that the current rule incorrectly classifies. The Uncovered Examples Window lists all examples that belong to the class indicated by the rule's conclusion, but which are not covered by the rule. These windows indicate to the user the evidence that led the system to select the current rule. They can be used both to aid the user's understanding of the underlying support for the rule and to help the user to formulate alternatives to the rule that serve the same purpose. They also provide a simple mechanism for the user to ask *what if?* and *why not?* questions. To determine why the system did not select a specific alternative to a rule, the user need only create and select the other candidate. If it was not created because it failed to cover specific examples, this will be indicated by the contents of the Positive Examples Window. If the failure to create it was due to the alternative rule incorrectly classifying

Fig 2 about here

specific cases, this will be indicated by the contents of the Counter Examples Window. Finally, they provide a mechanism whereby the machine learning system can critique rules that the user formulates. If a rule fails to cover example cases, this will be indicated by the Uncovered Examples Window. If a rule misclassifies cases, this will be indicated by the Counter Examples Window. Finally, if the rule performs well, this will be indicated by the Positive Examples Window. The operation of these windows is illustrated in Fig. 2. Here, a rule for diagnosing Immunoglobulin A Deficiency (IGA_NX) is selected. This rule has been learned by the machine learning system. The Positive Examples Window displays the examples that support the rule. The Uncovered Examples Window displays the IGA_NX cases that the rule fails to cover. The empty Counter Examples Window shows that this rule does not misclassify any examples.

Fig 3 about here

To illustrate the use of these windows to answer *why not?* questions, Fig. 3 displays the same situation after the user deletes the first clause from the rule. This is equivalent to asking why not develop this alternative rule? As can be seen from the Counter Examples Window, the answer is because the alternative rule covers the two cases that are listed.

A fifth example window also relates to the current rule. The Insufficient Information Examples Window lists any cases for which it cannot be determined whether or not the rule covers the example. This will occur when an example has unknown values for some of the attributes referred to in a rule, and satisfies all clauses relating to attributes for which it has values defined. This window alerts the user to potential problems arising from missing values in the data.

The previous four example windows relate to the performance of a rule considered in isolation. The remaining two example windows relate to the performance of a set of rules as a whole. The Misclassified Examples Window lists cases that are misclassified by the rule set. Such misclassification may occur even if the example is correctly classified by one or more rules, if it is also misclassified by other rules. The manner in which such conflicts between rules are handled by the system is under user control. Rules may be ordered, so that the first rule encountered that covers a case is used. Alternatively, rules may be assigned values. Under this treatment, of the rules that cover a case, that with the highest value is employed. Rule values may be specified by the learning system or by the user. A final alternative is that in cases of multiple conflicting conclusions, the case remains unclassified.

The Unclassified Examples Window lists cases that are not classified by the rule set. This may occur if multiple rules cover a case, as described above, or if no rule covers a case. The system may be set so that if no rule covers a case either the case is unclassified or a default classification is assigned.

These two windows allow the user to obtain holistic evaluation of the rule set in addition to the rule specific evaluation provided by the majority of examples windows.

The examples windows serve another function—example set validation. It is very easy for errors to enter into data. These may occur through errors in data entry or transmission or even due to imprecision in measuring instruments. In the traditional machine learning context, such errors will only rarely become apparent, as the knowledge bases that are inferred are not usually related back to the training examples. In contrast, The Knowledge Factory's examples windows encourage the

user to closely inspect significant examples. In particular, the Insufficient Evidence, Counter, and Uncovered Examples Windows will often highlight cases that contain errors.

Case-based rule critique

The example windows enable the machine learning system to justify its actions to the user and to critique rules created by the user. The user is also able to critique rules formed by the learning system. In our experience it is common that experts will dislike a rule formed by the learning system but be unable to articulate appropriate modifications thereto. In such circumstances the expert will often be able to provide examples to support his or her disquiet. It is straightforward to add those examples to the training set. Such examples may be complete, including values for all attributes, and possibly derived from an actual case from the expert's experience. Alternatively, they might be partial, with values specified only for some attributes. Such an example provides an abstract characterisation of types of case for which an alternative decision is appropriate.

Case-based rule editing

Another manner of using examples to modify rules is provided by case-based rule editing. Specification of counter-examples provides the machine learning system with information about conditions under which a rule should not apply. This may either cause the system to specialise a rule by adding more constraints to its application, or to select a different set of attributes with which to condition the application of the rule. To extend a rule to cover additional types of situation is less straightforward however. The addition of further positive examples to the training set may result in a rule being modified to cover those cases when learning is next applied. However, the system may equally cover those cases through modification of other rules for the class, or the generation of new rules. Also, depending on the metric used to evaluate the quality of rules, the machine learning system may tolerate a small number of counter-examples to a rule due to the quality metric evaluating them as not sufficient to warrant specialising the rule, thus defeating the user's objective in specifying a counter-example.

Case based rule editing has been developed to deal with such circumstances when an expert is unable to directly specify suitable changes to a rule, but can identify appropriate counter-examples or uncovered positive examples. With this technique, the expert identifies a rule and a case that the rule should or should not cover, and the system suggests modifications to the rule that would produce the desired outcome. The user can then evaluate and select between the proffered modifications or explore other forms of revision.

Fig 4 about here

When excluding a case from the cover of a rule, the system generates all least specialisations of the rule with respect to the case. This is illustrated in Fig. 4. In this example, we have removed a clause from a rule, which is highlighted in the Rules Window. This has caused it to incorrectly cover some counter examples, which are displayed in the Counter Examples Window. To explore how these counter examples might be excluded from the cover of the rule we have then applied the Contract to Exclude Case command, selecting the first case in the Counter Examples Window. This generates a large number of alternative rules, of which the last three are displayed in the Alternative Rules Window. Two of these add possible constraints on the

attribute *age* while the last adds a constraint on the attribute *gender*. The effect of these new constraints is summarised on the summary line after each rule and could be explored further using the examples windows. Further case based editing could be applied to one of these alternatives until a satisfactory outcome was obtained, or the user might reach a stage where he or she is able to complete the necessary modifications directly.

When generalising a rule to cover a case, the system generates the least generalisation of the rule with respect to the case. For the types of attribute-value rule supported, there will only ever be a single least generalisation of a rule with respect to a single case. If a user specifies a new case that a rule should cover, the Expand to Cover Case command can be used to force the system to immediately generalise that rule to cover that case. As with the Contract to Exclude Case command, several such case-based rule editing actions may be required before a user is satisfied with the resulting rule.

Machine generated examples are not used

It is interesting to contrast the case-based communication mechanisms utilised by The Knowledge Factory with the techniques pioneered in MARVIN (Sammut & Banerji, 1986). When testing hypotheses, MARVIN actively generates 'examples' which it asks the expert to classify. Such a mechanism has been deliberately excluded from The Knowledge Factory in the belief that, for many domains, without access to the types of background knowledge that we have also deliberately excluded from the system, examples created by the computer are likely to be nonsensical. In a medical domain, for example, the interactions between and constraints on attributes will be many and varied. Computer generated cases are likely to violate such constraints, for example, by hypothesising pregnant male patients. The exploration of violation of such constraints is unlikely to be useful, as the expert will know that the possibility does not have to be excluded from a rule as it will never arise in practice.

Our case-based communication mechanisms exclusively use cases specified by the user or imported from outside the system. In consequence, all cases should be meaningful to the user. The computer may present cases to the user, but only cases selected from those provided to it.

Exploration of Alternative Rules

Case-based rule editing is able to assist in some circumstances where an expert 'knows' that a rule is unsuitable, but cannot identify precisely how. But, it is limited to situations in which the user can specify or identify appropriate example cases. This will not always be possible. Alternative rules are another mechanism that The Knowledge Factory provides for such situations.

Machine learning systems frequently make arbitrary choices between possible tests for inclusion in a rule. There will often be several tests, all of which cover exactly the same set of training cases. A machine learning system must select one of these, and usually does so in an arbitrary manner. That such a choice has been made will not usually be flagged to the user. However, while the choice may have been arbitrary given the information available to the system, an expert may well be able to make comparative qualitative judgements about the alternatives. This problem could be tackled by involving the user in the process of selecting tests during induction. However, this would impose a large burden on the user. There are very many such tests considered

in typical learning contexts. Further, in many situations the user may not have any better basis on which to choose between alternatives than does the system.

Instead, The Knowledge Factory offers the user a facility for identifying and selecting between alternative rules. This facility can be invoked by the user at any time at his own discretion. The Form Alternative Rules command creates a set of alternatives to the current rule. First, the system identifies all most general rules that cover all cases correctly covered by the current rule. Two more rules are added. The first is the most specific rule to cover all of the cases correctly covered by the initial rule. This provides for the user a complete presentation of all tests that could possibly be used to cover these cases. The second additional rule has as its antecedent the conjunction of all antecedents of all the most general rules that were formed. This summarises for the user the set of all tests that appear to be effective in distinguishing the current class from others. These rules are displayed in a window within which the user may explore them using all of the system's rule editing and evaluation facilities.

Fig 5 about here

Figure 5 illustrates the outcome of this command. In this example, rules have been developed from examples of diagnosis of hypothyroid disease (Quinlan, Compton, Horn, & Lazarus, 1986). The rule highlighted in the Ruleset Window is the original rule for which alternatives have been developed. The Alternative Rules Window lists the four alternatives that were discovered. The last two of these rules are the most general alternatives. The first of these is the original rule. The second is identical to the first except that the clause *psych is f* is replaced by *sick is f*. All 43 cases covered by the original rule satisfy both of these conditions. One or the other of these clauses is needed if a rule is to cover all 43 cases correctly classified by the original rule and no cases of another class. For this rule there happens to be one case of class *compensated_hypothyroid* that satisfies all of the other conditions but, unlike any of the positive cases, has values of *t* for both *psych* and *sick*. This is a clear illustration of a situation in which system has no sensible basis on which to choose between alternatives but an expert may. The first of the alternative rules is the most specific rule to cover all 43 cases. The second alternative rule contains all clauses in either of the most general alternatives.

Fig 6 about here

Note that this is a very simple example of alternative rules, deliberately chosen for ease of presentation and explanation. For contrast, Fig. 6 presents some of the 39 alternative most general rules all of which cover the only two cases of class secondary_hypothroid in the training set and none of which cover any cases of another class.

There is an interesting relationship between this alternative rules technique and the example generation technique of MARVIN (Sammut & Banerji, 1986). In both situations the system wishes to obtain advice from an expert to help it distinguish between alternatives that are equally well supported by the training cases. However, whereas MARVIN generates hypothetical cases in an attempt to distinguish between the alternatives, we present the alternatives directly. As argued above, automated generation of hypothetical cases will often be unsuccessful in real world domains, if only because such cases are unlikely to appear realistic to an expert. We believe that complete rules of the type that The Knowledge Factory creates will often be easier for the expert to comprehend and manipulate.

Interactive Domain Model Revision

Buntine & Stirling (1991) argue that effective real world application of machine learning frequently requires a cyclical process in which the results of machine learning are provided to the expert for review which may lead to the identification of relevant information not originally provided to the machine learning system, which leads back to another phase of application of machine learning with revised inputs. One of the design objectives for The Knowledge Factory is to allow the expert to complete this cycle within the system. With many machine learning systems, each time through the cycle, machine learning must start afresh. I have already described how the system can iterate through such a cycle when the new information is presented in the form of new or revised example cases. Greater change is required, however, if the expert realises that there are deficiencies in the set of attributes selected with which to describe the examples. We have observed this circumstance in practice. The expert, when confronted with a rule developed by the system, and the evidence it can provide in the rule's support, realises that a factor not included in the current case descriptions is necessary to adequately revise the rule. Another type of change is required when it is realised that there is a need to provide finer grade distinctions for an existing attribute (one or more current value needs to be replaced by a number of values) or some other transformation of an attribute is needed (conversion from ordinal to categorical or vice versa, for example). The Knowledge Factory provides facilities for performing such conversions. Existing rules and cases that are affected by the change are automatically transformed, where possible, so as to retain as much as possible of the existing knowledge base.

When a new attribute is created, all cases are initialised with the value for this attribute set to *unknown*. When the attribute is only relevant in a small number of contexts, the user need only specify values for a small number of cases, for example, the positive and counter examples for the current rule.

Direct Expert Evaluation of Rules

While cased based communication is able to accommodate many of the interactions between an expert and a machine learning system, there are some communication needs that it cannot cover. One of these is the direct communication of general qualitative rule evaluation by the expert. Case-based communication allows the expert to inform the system about specific deficiencies of a rule. It does not allow the expert to specify that the current form of a rule is very good, or that it is unacceptable without specifying how.

The Knowledge Factory allows the expert to attach qualitative labels to individual rules. The three labels currently supported are *no good*, *revisable*, and *good*.

The Knowledge Factory takes account of these assessments during induction. Rules that are rated *no good* are discarded before rule refinement commences. This usually results in the formation of totally new rules, although in some circumstances the induction system may derive the same rule again. Techniques for avoiding this outcome would be desirable, but are difficult to incorporate into traditional machine learning systems.

Rules that are rated *good* are not altered during rule refinement, but rather are passed directly into the final rule set.

Revisable rules are retained, but may be altered to reduce the number of negative cases or increase the number of positive cases covered. This is the default annotation.

These rule annotations are simple for the user to utilise, but provide useful guidance to the rule refinement process.

Summary Ruleset Evaluation

Most machine learning systems provide facilities for summarising the performance of inferred rules when applied to an evaluation set of cases. The primary metric used in such evaluation is usually accuracy. But in real world applications, this will not always be the most important measure of performance. In many domains, some errors are more significant than others (for example, failure to diagnose a serious disease in contrast to unnecessary but comparatively harmless treatment arising from diagnosing a disease that is not present). We have sought to develop a simple manner in which to capture all of the relevant summary information that is likely to be useful for a user.

Fig 7 about here

Figure 7 presents an example of the tabular format for presenting evaluation results that we have created. As there are four classes in this example (using the hypothyroid diagnosis data previously mentioned), there is a 4x4 matrix which presents the number of cases belonging to each class that have been given each classification. The first four rows of the table relate to the class assigned to a case by the current rule set while the first four columns relate to the correct class for a case. The fifth column relates to all cases. The sixth column presents for each class, the percentage of those cases assigned that class by the rule set for which that classification was correct. The fifth row presents the total number of cases of each class for which any classification was assigned by the rules. The sixth row summarises the percentage of those that were correct. The seventh row indicates the number of cases for each class for which no decision was made. The eighth row shows the proportion of all cases for the class for which a classification was assigned. The ninth row displays the sensitivity of the rule set for the class. This is the proportion of cases belonging to the class that were correctly classified. The final row displays the specificity for the class. This is the proportion of cases not belonging to the class that were not incorrectly classified as belonging to the class.

EXPERIMENTAL EVALUATION

A major difficulty facing research in knowledge acquisition is evaluation of alternative techniques. Real world knowledge acquisition tasks usually require large and very expensive projects. It is not feasible to conduct controlled experiments in which different methodologies are each applied to the same full scale knowledge acquisition project under real-world conditions. As a result, other than case studies, there has been little experimental investigation of the performance of different techniques. Even the ground-breaking Sisyphus projects (Linster, 1992; Schreiber & Birmingham, 1996), in which many different systems were all applied to common knowledge acquisition tasks, is perhaps best thought of as a set of parallel case studies, as it was not possible to control extraneous factors that may have affected a participant's use of a system, and hence comparisons are restricted.

A case study can demonstrate the ability of a technique to achieve an outcome. Beyond this, they provide little comparative information about alternatives.

We have been concerned to perform experimental evaluation of the efficacy of the techniques embodied in The Knowledge Factory. We were encouraged by the positive evaluations provided by participants in case studies (Webb, 1996).

We wanted to have large numbers of subjects apply differing approaches to a knowledge acquisition task under controlled conditions. One way to achieve this was to set appropriate assignments for students in an undergraduate course on Artificial Intelligence and Expert Systems, and to study their performance. Subjects could be provided software with different capabilities and their performance measured on common tasks under controlled conditions.

An initial attempt at such evaluation (Webb & Wells, 1996) was undermined by a user interface flaw. Many subjects seeking to use The Knowledge Factory's machine learning facilities to refine rules that they had created unintentionally employed machine learning in a mode that deleted their rules and replaced them by rules learned by the machine learning system independently from the initial rules. In consequence, while on average outperforming both knowledge acquisition without access to machine learning facilities and the application of machine learning alone, the integrated application of machine learning with knowledge acquisition from experts failed to demonstrate a statistically significant advantage over machine learning alone. When the subjects that applied machine learning in the mode that overwrote existing rules were excluded from consideration, the advantage over machine learning was statistically significant, but the selection of subjects for analysis in this manner cast doubt upon the results.

This user interface fault was rectified, and a second such experiment undertaken (Webb, Wells, & Zheng, 1999). In this second experiment, we again used undergraduate students performing a class assignment. This assignment was conducted under supervised laboratory conditions. Knowledge acquisition tasks were created using the Glass and Soybean Large data sets from the UCI Repository of Machine Learning Databases (Merz & Murphy, 1997). The Glass data relates to forensic analysis of glass fragments and the Soybean Large data pertains to the diagnosis of soybean plant diseases.

Subjects were given randomly selected training sets of examples. The accuracy of their expert systems could then be tested by evaluating their performance on withheld evaluation sets. To simulate use by experts, the subjects were provided with rules for these tasks learned by C4.5rules (Quinlan, 1993) from subsets of the data. As these subsets included both training and evaluation cases provided to the subjects, the rules could be expected to capture information about the domains that subjects could not glean from examination of their training data either with or without the use of machine learning tools.

Two versions of The Knowledge Factory were created. A number of facilities were disabled that could not be usefully employed in the context of the study. This was to minimise the chance of a repetition of the previous outcome where misapplication of the software invalidated the study. The facilities disabled included changing the manner in which rules were interpreted during execution of a rule set, saving and loading sets of examples from disk, and the selection, maintenance and utilisation of separate training and evaluation sets. All of the key knowledge acquisition facilities were retained, except that machine learning capabilities were disabled in one version, called *KA-alone*.

Subjects were randomly divided into two groups. One group used KA-alone for the Glass data and the full system for Soybean Large, and the other group used the systems in the other order.

Table 1 about here

Seventeen subjects participated in the experiment. Table 1 presents the mean predictive accuracy on the evaluation cases obtained using each system for each data set, along with that of The Knowledge Factory's machine learning system when applied using the training and evaluation cases used by subjects using the full system. For both data sets the integrated use of machine learning with knowledge acquisition from experts led to higher predictive accuracy than the use of either of its constituent approaches in isolation. With respect to KA-alone, both of these differences were statistically significant at the 0.05 level (one-tailed two-sample t tests; Soybean Large:t=2.1, p=0.026; Glass: t=3.35, p=0.001). With respect to learning-alone, the difference was significant at the 0.05 level for the Glass data (one-tailed matched-pairs t test: t=2.5, p=0.021) but the other was not (one-tailed matched-pairs t test: t=0.9, p=0.279). Note that the power of these tests was low due to the small number of subjects and hence that the failure to find a significant difference provides only very weak evidence that such a difference does not actually exist. The significant differences that were found demonstrate that for at least some knowledge acquisition tasks, the integrated use of machine learning with knowledge acquisition from experts can produce more accurate expert systems than either constituent method in isolation.

The integrated approach was also faster than KA-alone. For Soybean Large the mean and standard deviation for the integrated approach was 73±45 minutes while for KA-alone it was 131±19. For Glass these figures were respectively 16±19 and 115±38. One-tailed two-sample t tests reveal that both of these differences are significant at the 0.05 level (Soybean Large: t=3.3, p=0.002; Glass: t=1.9, p=0.037). In both cases the time savings from the integrated use of machine learning with knowledge acquisition were substantial. Of course, they were both eclipsed in this respect by the application of machine learning alone, for which knowledge acquisition time can be measured in seconds rather than minutes.

Questionnaire responses from participants indicated that the subjects believed the machine learning facilities were useful, found the knowledge acquisition process easier when machine learning facilities were available, and had greater confidence in the expert systems developed with the aid of machine learning.

These results, presented and analysed in more detail by Webb, Wells, & Zheng (1999), all provide support for the efficacy of the approaches that we have developed. At least for some knowledge acquisition tasks they can lead to the more rapid development of more accurate expert systems in which the users have greater confidence than does knowledge acquisition without machine learning, and to more accurate expert systems than machine learning in isolation.

CONCLUSIONS

The Knowledge Factory is a knowledge acquisition system that is designed for direct use by domain experts. It differs from most previous systems intended for this use by incorporating machine learning facilities. In consequence, it is not necessary for the expert to provide solutions for every contingency that the knowledge base must cover. The machine learning system can fill gaps in the expert's expertise. Both the machine

learning system and the expert can critique and propose refinements to the other's rules.

One of the distinctive features of The Knowledge Factory is the manner in which communication with the user has been structured around the use of example cases. Techniques have been developed that allow both the user and the system to convey non-trivial information through this mechanism. Interestingly, however, we have concluded that one such form of communication that has received previous use, the construction of hypothetical cases by the system in order to test hypotheses, is actually inappropriate in our context. Instead, we favour explicit presentation of the competing hypotheses in this situation.

The Knowledge Factory is distinguished from previous knowledge acquisition systems by the manner in which it supports experts with minimal computing expertise to directly interact with a machine learning system during all phases of knowledge acquisition. Case studies have found that such users have little difficulty in using the system and controlled studies suggest that the integration of machine learning with knowledge acquisition within the system can outperform either constituent approach in isolation.

REFERENCES

Agar, J., & Webb, G. (1992) The application of machine learning to a renal biopsy data-base. *Nephrology, Dialysis and Transplantation*, **7**: 472-478.

Attar Software (1989). *Structured decision tasks methodology for developing and integrating knowledge base systems*. Attar Software, Leigh, Lancashire.

Boose, J. H. (1986). ETS: A system for the transfer of human expertise. In J. S. Kowalik (Ed.), *Knowledge based problem solving*. New York: Prentice-Hall.

Buntine, W., & Stirling, D. (1991). Interactive induction. In J. E. Hayes, D. Michie, & E. Tyugu (Eds.) *Machine Intelligence 12*. Oxford: Clarendon Press, pp. 121-137.

Compton, P., Edwards, G., Srinivasan, A., Malor, R., Preston, P., Kang, B., & Lazarus, L. (1992). Ripple down rules: Turning knowledge acquisition into knowledge maintenance. *Artificial Intelligence in Medicine*, **4:** 47–59.

Davis, R., & Lenat, D. B. (1982). *Knowledge-based systems in artificial intelligence*. New York: McGraw-Hill.

De Raedt, L. (1992). *Interactive theory revision*. London: Academic Press.

Kodratoff, Y., & Vrain, C. (1993) Acquiring first-order knowledge about air traffic control. *Knowledge Acquisition*, **5**, 1–6.

Linster, M. (1992) A review of Sisyphus 91 and 92: Models of Problem-Solving Knowledge. In N. Aussenac, G. Boy, B. Gaines, M. Linser, J.-G. Ganascia, & Y. Kordratoff (Eds) *Knowledge Acquisition for Knowledge-Based Systems*. Berlin: Springer-Verlag, pp. 159-182.

Merz, C. J. & Murphy, P. M. (1997) *UCI Repository of Machine Learning Databases* [Machine-readable data repository]. University of California.

Michalski, R. S. (1983). A theory and methodology of inductive learning. In R. S. Michalski, J. G. Carbonell, & T. M. Mitchell (Eds.) *Machine learning: An Artificial Intelligence Approach*. Berlin: Springer-Verlag.

Morik, K., Wrobel, S., Kietz, J.-U., & Emde, W. (1993). *Knowledge Acquisition and Machine Learning: Theory, Methods, and Applications*. London: Academic Press.

Nedellec, C., & Causse, K. (1992). Knowledge refinement using knowledge acquisition and machine learning methods. *Proceedings EKAW'92*. Berlin: Springer-Verlag, pp. 171-190.

O'Neil, J. L., & Pearson, R. A. (1987). A development environment for inductive learning systems. In *Proceedings of the 1987 Australian Joint Artificial Intelligence Conference*. Sydney, pp. 673-680.

Plotkin, Gordon D. (1970) A note on inductive generalisation. In B. Meltzer & D. Mitchie (Eds) *Machine Intelligence 5*. Edinburgh University Press, Edinburgh, pp. 153-163.

Quinlan, J. R. (1993). *C4.5: Programs for Machine Learning*. San Mateo, CA: Morgan Kaufmann.

Quinlan, J. R., Compton, P., Horn, K. A., & Lazarus, L. (1986) *Inductive Knowledge Acquisition: A Case Study*, New South Wales Institute of Technology School of Computing Sciences, Technical Report 86.4, Sydney.

Sammut, C. & Banerji, R. B.(1986) Learning concepts by asking questions. In Michalski, Ryszard S., Carbonell, Jaime G., & Mitchell, Tom M. (Eds) *Machine Learning: An Artificial Intelligence Approach Volume II*. Morgan Kaufmann, Los Altos, pp. 167-191.

Schmalhofer, F., & Tschaitschian, B. (1995). Cooperative knowledge evolution for complex domains. In G. Tecuci, & Y. Kodratoff (Eds.) *Machine learning and knowledge acquisition: Integrated approaches*. London: Academic Press.

Schreiber, A. & Birmingham , W. P. (1996) The Sisyphus-VT initiative. *International Journal of Human-Computer Studies*. **44**.

Shapiro, A. (1987) Structured Induction in Expert Systems. Addison-Wesley, London.

Smith, R. G., Winston, H. A., Mitchell, T. M., & Buchanan, B. G. (1985). Representation and use of explicit justifications for knowledge base refinement. In *Proceedings of the Ninth International Joint Conference on Artificial Intelligence*. San Mateo, Ca: Morgan Kaufmann, pp. 673-680.

Tecuci, G., & Kodratoff, Y. (1990). Apprenticeship learning in imperfect domain theories. In Y. Kodratoff, & R. Michalski (Eds.) Machine learning: An Artificial Intelligence Approach. San Mateo, CA: Morgan Kaufmann.

Webb, G. I. (1993). DLGref2: Techniques for inductive knowledge refinement. In *Proceedings of the IJCAI Workshop W16*. Chambery, France, pp. 236-252.

Webb, G. I. (1996). Integrating machine learning with knowledge acquisition through direct interaction with domain experts. Knowledge-Based Systems, **9**: 253-266.

Webb, G. I. & Wells, J. (1996) Experimental evaluation of integrating machine learning with knowledge acquisition through direct interaction with domain experts. In P. Compton, R. Mizoguchi, H. Motada, & T. Menzies (Eds) *Proceedings of PKAW'96: The Pacific Knowledge Acquisition Workshop*. Sydney, pp. 170-189.

Webb, G. I., Wells, J., & Zheng, Z. (1999) An experimental evaluation of integrating machine learning with knowledge acquisition. *Machine Learning*, **35**(1): 5-24.

Wilkins, D. C. (1988). Knowledge base refinement using apprenticeship learning techniques. In Proceedings of the Seventh National Conference on Artificial Intelligence. San Mateo, CA: Morgan Kaufmann, pp. 646-651.

APPENDIX A

THE DLGREF ALGORITHM

α is the most specific possible rule for the class positive, that covers no cases. The least generalisation of α against a case results in the most specific rule that covers that case.

ζ is the most general possible rule for the class positive, that covers all cases.

Function DLGref

Parameters: *rules*: an initial set of rules for a single class
POS: a set of cases belonging to that class
NEG: a set of cases that do not belong to the class
value(): a function from rules to numeric values such that the higher the value the greater the preference for the rule.

Returns: *rules*: a revised set of rules for the class

```
for r is set to each rule in rules in succession
    if r covers any cases in NEG
        spec_rule <- generalise_rule(α, covered_cases(r, POS), NEG, value)
        if spec_rule covers any cases in POS
            r <- generalise_toward(spec_rule, r, NEG)
        end if
    end if
    remove from POS all cases that r covers
end for
for r is set to each rule in rules in succession
    r <- generalise_rule(r, POS, NEG, value)
    remove from POS all cases that r covers
end for
while POS is not empty
    r <- generalise_rule(α, POS, NEG, value)
    if r covers any cases in POS
        r <- generalise_toward(r, ζ, NEG)
        remove from POS all cases the r covers
        add r to rules
    else
        remove all remaining cases from POS
    end if
end while
```

Function generalise_rule

Parameters: *rule* an initial rule for the positive class
POS: a set of cases belonging to that class
NEG: a set of cases that do not belong to the class
value(): a function from rules to numeric values such that the higher the value the greater the preference for the rule.

Returns: *result*: a generalisation of *rule*

```
result <- rule
for c is set to each case in POS in succession
    set r to the least generalisation of result with respect to c
    if value(r, POS, NEG) > value(result, POS, NEG)
        result <- r
    end if
end for
```

Function generalise_toward

Parameters: *spec* an initial specific rule for the positive class
gen: a generalisation of *spec*
NEG: a set of cases that do not belong to the class

Returns: *result*: a rule that is a generalisation of *spec* and a specialisation of *gen* and which covers no cases in *NEG* not covered by *spec*

```
while spec ≠ gen
    for each clause c in the antecedent of spec
        if deleting c from spec would increase the number of cases in
        NEG covered by spec
            add c to gen
        end if
    end for
    for each clause c in the antecedent of spec
        if c is not in the condition of gen and adding c to the condition of
        gen does not decrease the number of cases in NEG covered by gen
            remove c from spec
        end if
    end for
end while
result <- spec
```

DLGref makes two passes through the initial rules. In the first pass, any rules that cover negative cases are specialised, if possible, so as to no longer cover those cases. All positive cases covered by the resulting rules are then deleted as they do not need further attention. In the second pass through the initial rules, they are generalised as much as possible to cover further positive cases. Any cases so covered are also deleted. Finally, new rules are added to the rule set to cover any remaining positive cases.

The function **generalise_rule** seeks to generalise an initial rule to cover as many as possible positive cases, but without unduly increasing negative cover. The value

function is used to evaluate whether an increase in negative cover, if it occurs, is sufficiently offset by an increase in positive cover.

The function **generalise_toward** generalises an initial specific rule toward a more general form as far as possible without increasing the negative cover of the initial rule.

Table 1: Mean and standard deviation for predictive accuracy.

Data set	Integrated	KA alone	Learning alone
Soybean Large	88.5±4.4	84.4±3.6	87.8±1.6
Glass	81.3±7.7	59.0±17.3	79.2±8.0

FIGURES

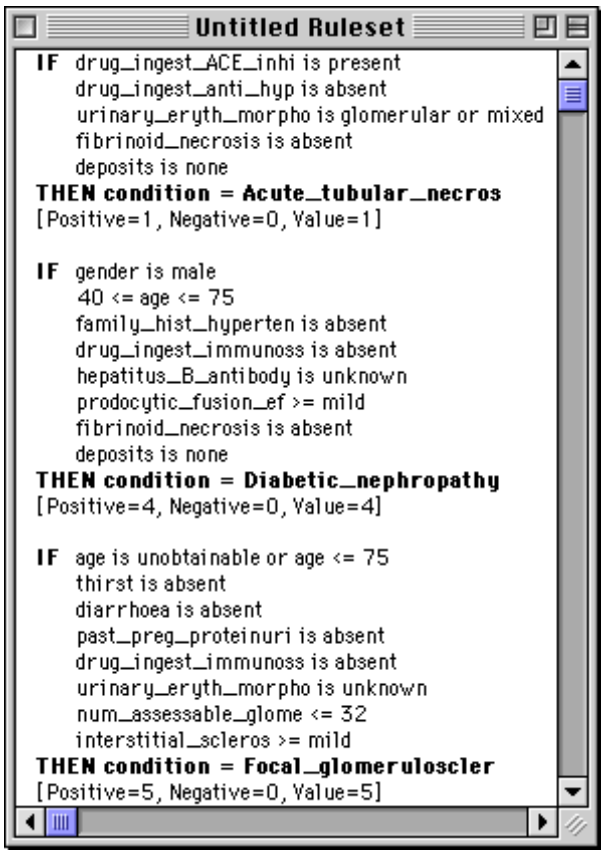

FIGURE 1: Examples of rules in The Knowledge Factory.

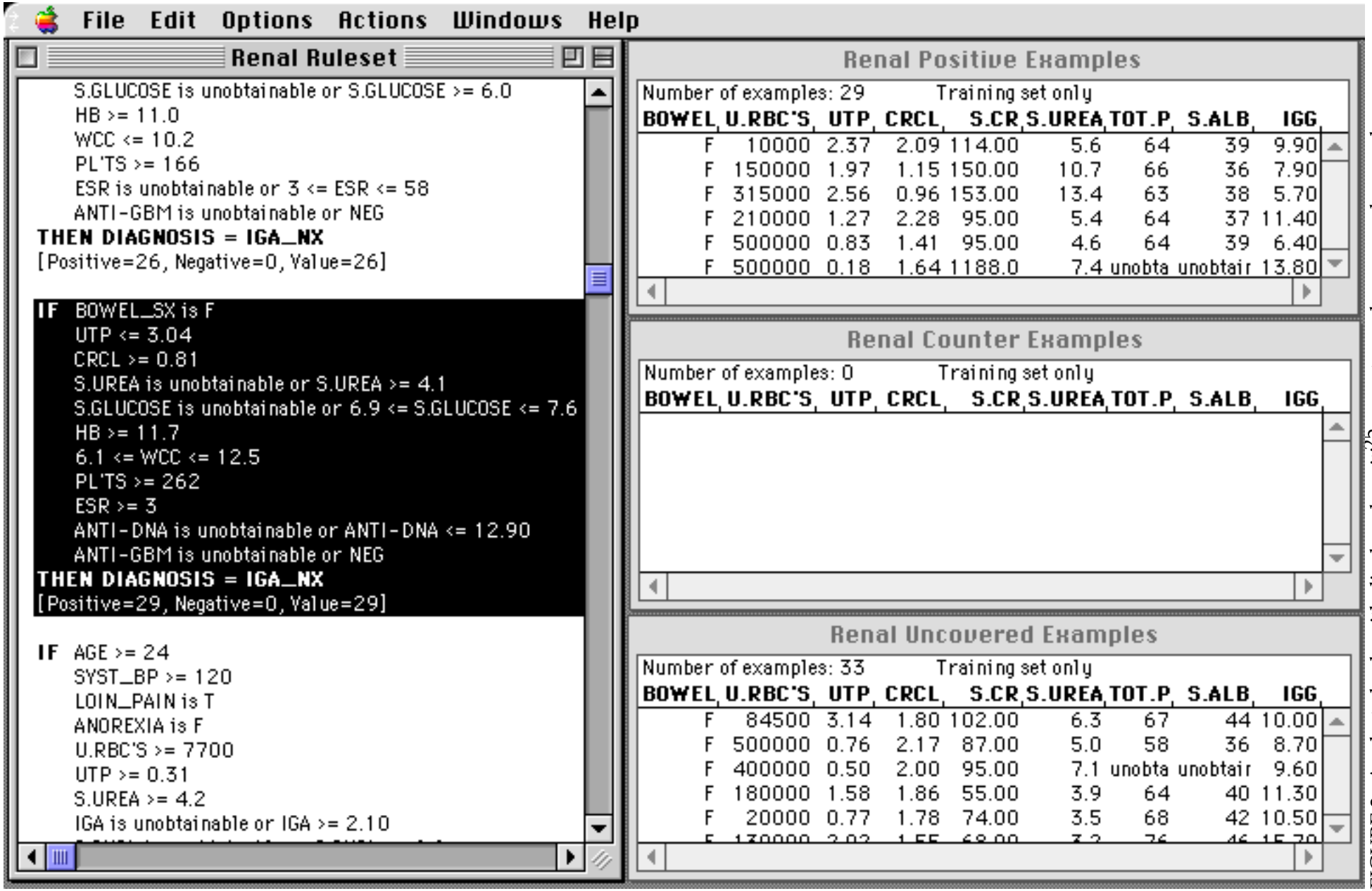

FIGURE 2: A selected rule with displayed positive, counter and uncovered examples.

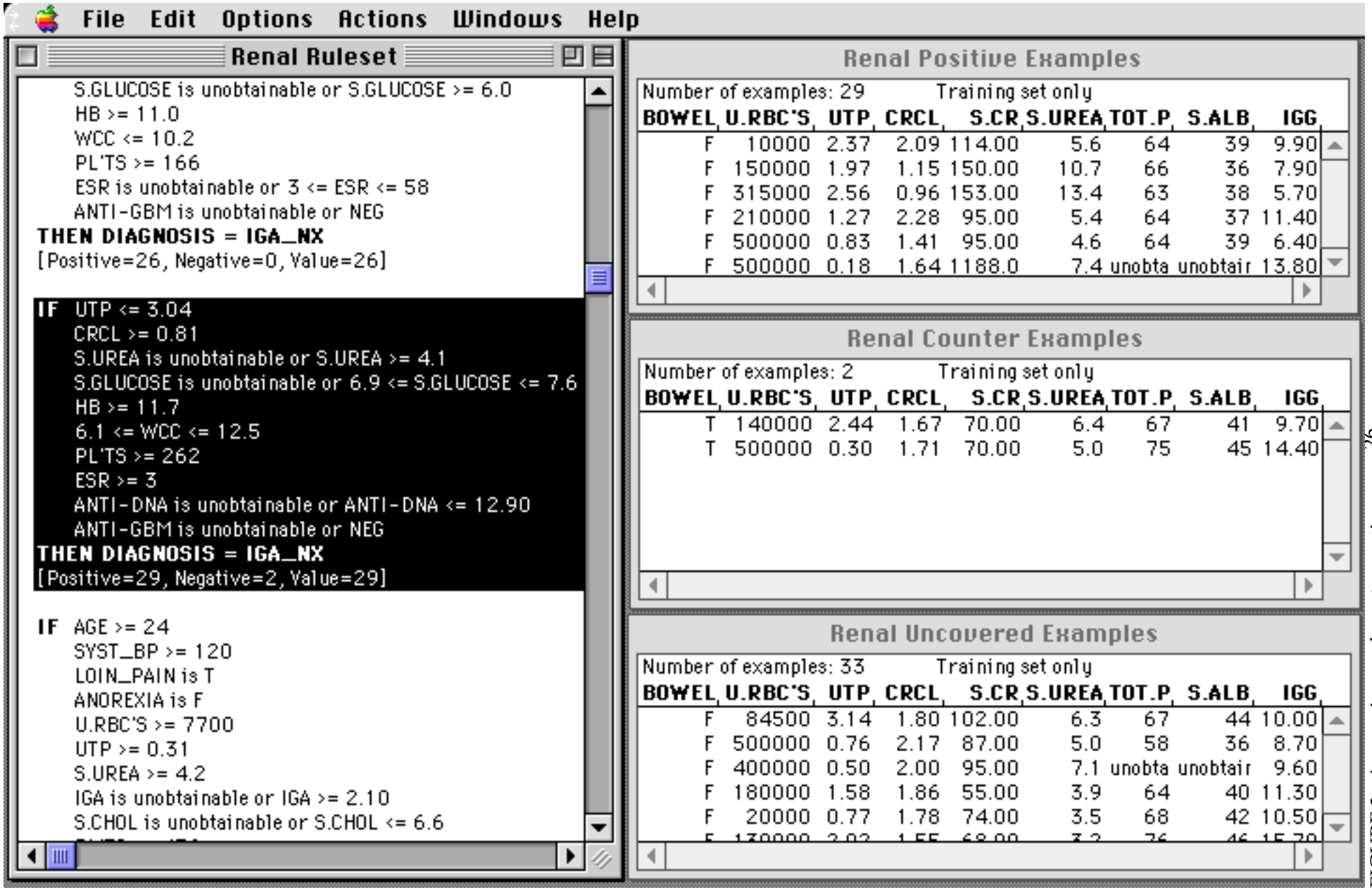

FIGURE 3: Answering a *why not* question.

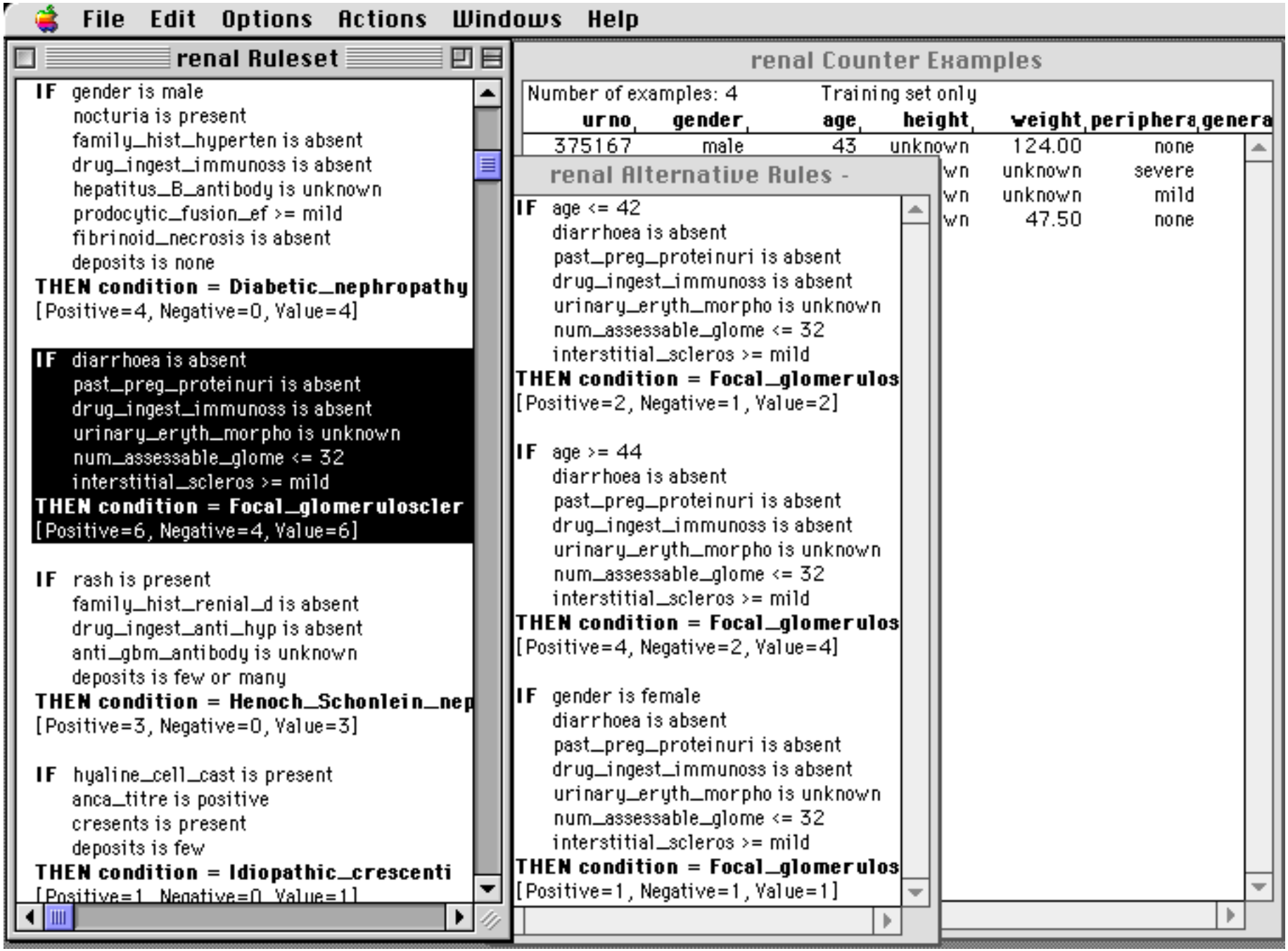

FIGURE 4: Case based editing: Rules generated by Contract to Exclude Case applied to the first case in the Counter Examples Window.

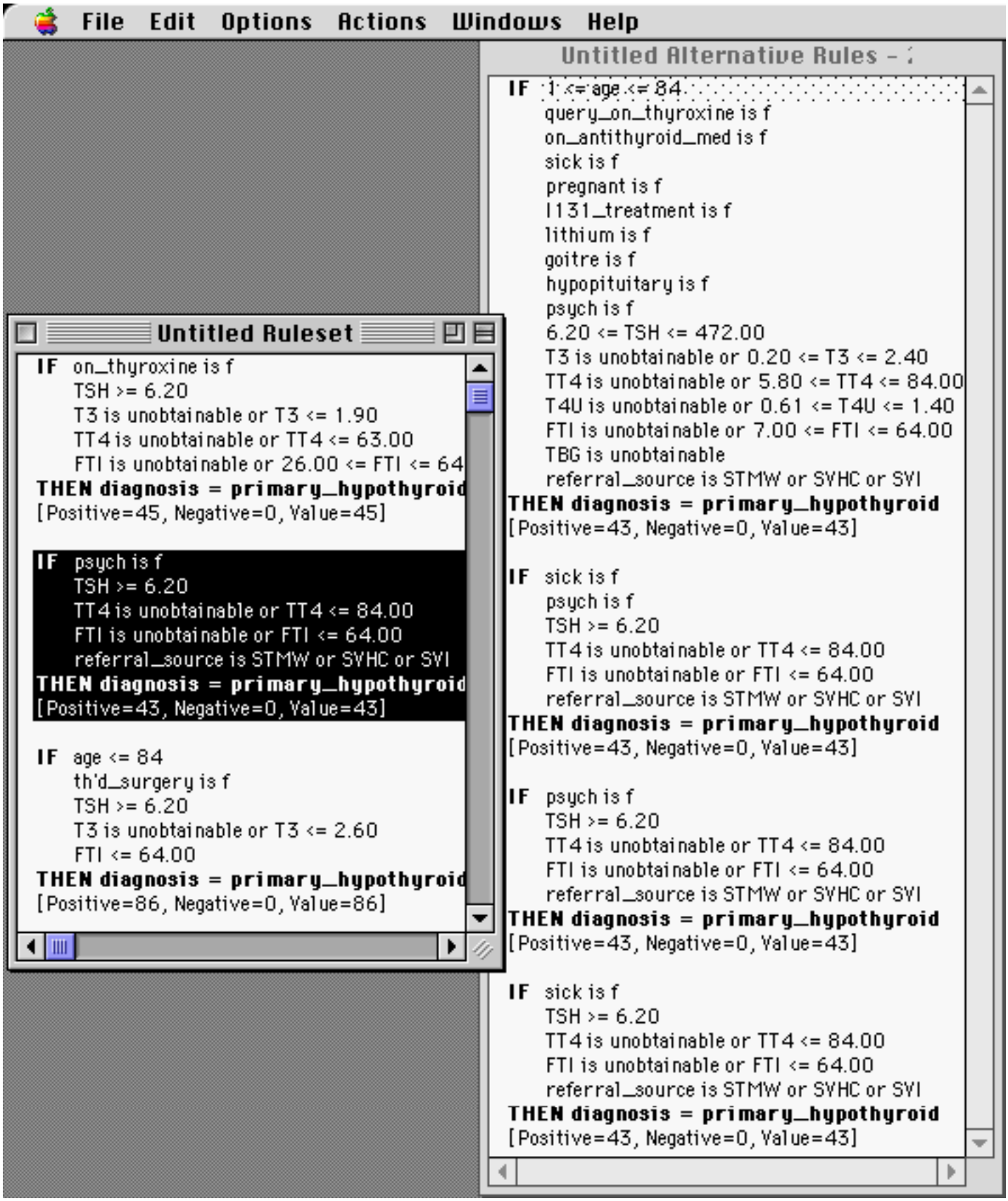

FIGURE 5: Alternative rules that cover the same 43 cases of class primary_hypothyroid.

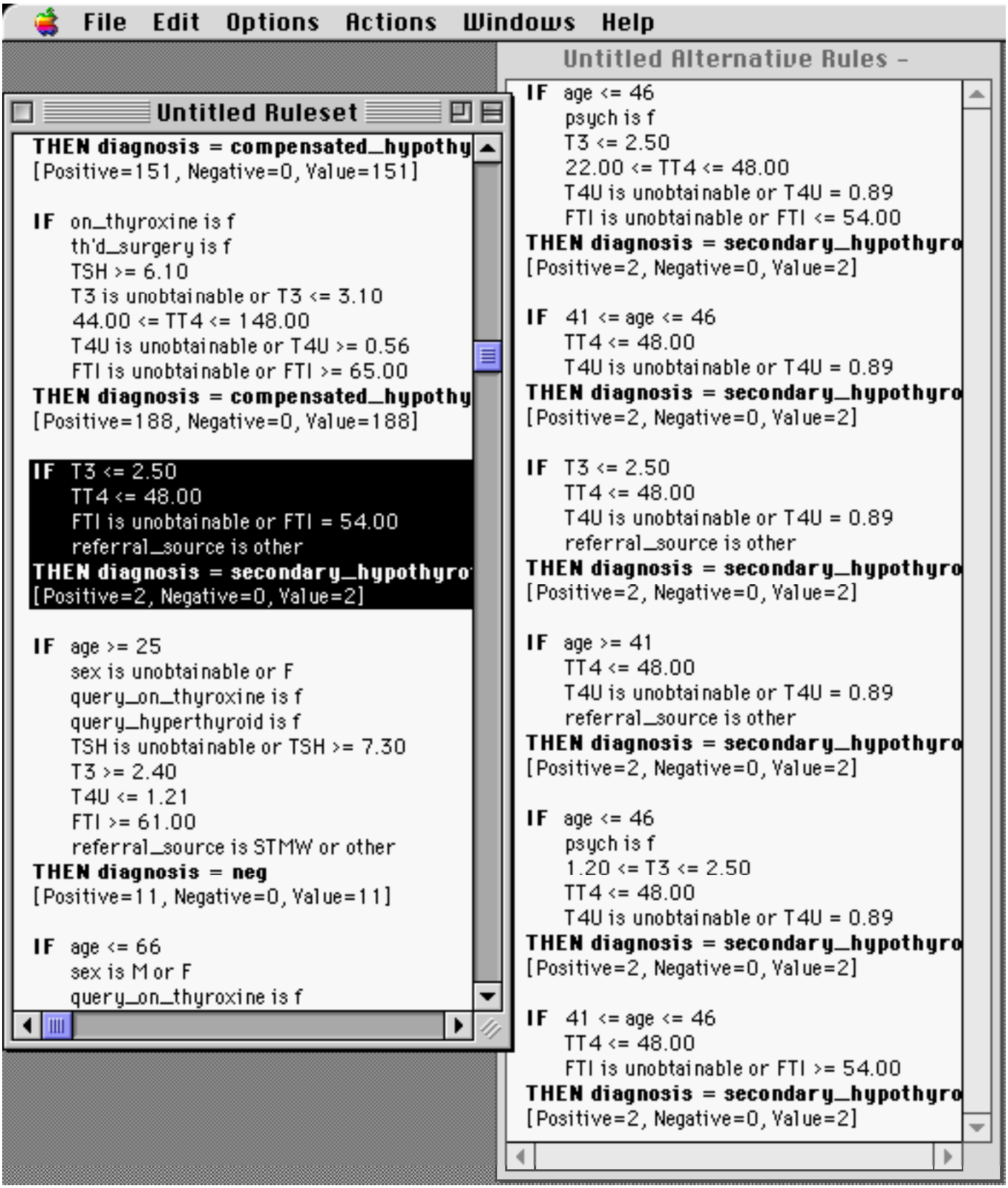

FIGURE 6: A selection of the 39 alternative most general rules that cover both cases of class secondary_hypothyroid.

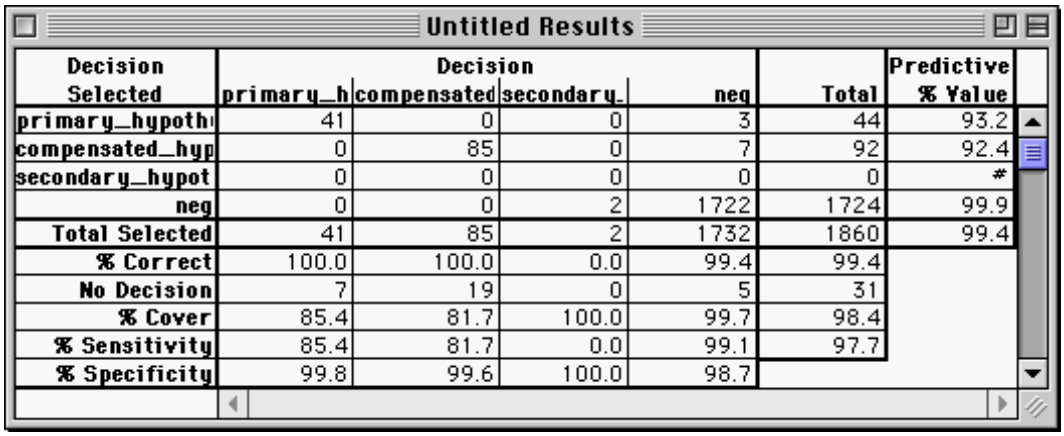

FIGURE 7: The results window.