Knowledge Acquisition with a Knowledge-Intensive Machine Learning System

Clifford A. Brunk and Michael J. Pazzani
Department of Information and Computer Science
University of California
Irvine, CA 92717
(brunk@ics.uci.edu)
(pazzani@ics.uci.edu)

Abstract

In this paper, we investigate the integration of knowledge acquisition and machine learning techniques. We argue that existing machine learning techniques can be made more useful as knowledge acquisition tools by allowing the expert to have greater control over and interaction with the learning process. We describe a number of extensions to FOCL (a multistrategy Horn-clause learning program) that have greatly enhanced its power as a knowledge acquisition tool, paying particular attention to the utility of maintaining a connection between a rule and the set of examples explained by the rule. The objective of this research is to make the modification of a domain theory analogous to the use of a spread sheet. A prototype knowledge acquisition tool, FOCL-1-2-3, has been constructed in order to evaluate the strengths and weaknesses of this approach.

1.0 Introduction

The emphasis of our research has been on the integration of machine learning and knowledge acquisition techniques to facilitate the refinement of rule-based expert systems. At one level, learning can be viewed as a process of translation from an input representation into a target representation. However, generally it is desirable that the learning program do more than faithfully record all of the information presented to it. The purpose of a learning algorithm is to generalize, distill or abstract the input information into a more useful form. From the perspective of knowledge acquisition, machine learning programs such as ID3 (Quinlan, 1986), EITHER (Ourston & Mooney, 1990)), KBANN (Shavlik & Towell, 1989), FOIL (Quinlan, 1990), GOLEM (Muggleton & Feng, 1990), FOCL (Pazzani & Kibler, 1992; Pazzani, Brunk & Silverstein, 1991), FORTE(Richards & Mooney, 1991), and AUDREY (Wogulis, 1991) are just black box abstraction machines. Given a collection of examples (and possibly some background knowledge) each algorithm outputs a description of these examples in an alternate representation- a decision tree, a set of rules, or a network of weights. Usually the only control one has over these algorithm comes in the form of example selection and parameter specification. This limited form of control requires that the user have a detailed understanding of the learning algorithm. The interactive learning system, CLINT (DeRaedt & Bruynooghe, 1989;1992), is an example of an algorithm that begins to open the learning process. Learning in CLINT is guided by the user's answers to system generated queries. The MOBAL (Morik, 1991) environment extends the level of cooperation between the user and the learning algorithm even further offering a balanced cooperative approach in which the user is free to enter any information that the system might use. The interaction between the user and the learner is not rigidly dictated by the algorithm. Beyond MOBAL, FOCL-1-2-3 offers a balanced cooperative approach to knowledge acquisition and allows the user more control over the learning algorithm than previous systems.

We do not advocate eliminating the knowledge engineer from the knowledge acquisition process; in the near term we do not believe it is realistic for an unaided domain expert to interact directly with a knowledge acquisition tool to create a knowledge base. However, the creation of knowledge acquisition tools that provide a medium for communication between the domain expert and the knowledge engineer will help the knowledge base creation process. The balanced cooperative approach to learning is a large step in the right direction in terms of integrating machine learning and knowledge acquisition techniques. However, we feel that it is equally important to open the learning algorithm itself- making it more accessible and easier to understand. FOCL-1-2-3 is an extension of the multistrategy first order Horn-clause learning program FOCL that incorporates both of these approaches. FOCL-1-2-3 extends FOCL in three significant ways. First, we have incorporated a graphic interface which allows dynamic visualization of the learning process. Second, we have integrated expert interaction into the low level learning algorithm by allowing the expert to select from a set of hypothesis. Third, we have extended the derivation structure that FOCL generates to include the examples that motivate each literal.

In addition to opening the learning process, we have added a number of features to FOCL that facilitate the knowledge acquisition process. The current generation of knowledge acquisition tools (e.g., Davis, 1978; Marcus, McDermott, & Wang, 1985; Wilkins, 1988) are useful in helping the user to discover why a single example is not classified correctly and in helping to identify a change to the knowledge base to correct this single example. However, they do not provide tools to assess the impact of this change on other examples that are correctly or incorrectly classified by the knowledge base. As a consequence, modifying a knowledge-base can be a time consuming activity in which changes in one rule produce unforeseen consequences that require changes in other rules. In contrast, the machine learning techniques we are employing use sets of examples to suggest revisions to the knowledge base. From the user's point of view, this technology simplifies the maintenance of the rule base of a knowledge-based system, by eliminating the iterative process of making a change based on a single example, and then discovering the impact of this change on other examples. In order to allow the user to manipulate and inspect a large numbers of examples in a coherent fashion, we have incorporated a spread sheet style interface.

The objective of this research is to make the modification of a domain theory analogous to the use of a spread sheet. That is, a change in the knowledge base should automatically update summary statistics describing which examples in a database are correctly classified by the knowledge base. Under user control, the specific examples that are affected by the change may be displayed. Furthermore, a change to the database should automatically update information on the accuracy and coverage of rules in the knowledge base. In addition (and unlike a spread sheet), changes to the database should result in the automatic creation of new rules (subject to expert inspection and confirmation) that abstract regularities from the data. Towards this end FOCL-1-2-3 maintains a close connection between examples and associated rules. Figure 1 illustrates how this information is currently made available to the user. The display indicated that eligible_for_deferment is true of eighteen people who are not required to pay back on their student loan and two people who are. The user has requested to see the examples which satisfy student_deferment, the rule responsible for covering the two negative examples. From this information, one can determine that three of the six students not required to pay back their loans are enrolled in 12 units, while the remaining students not required to pay back their loans are either enlisted in the armed forces (and thus eligible for a military deferment) or disabled (and thus eligible for a disability deferment). Furthermore, none

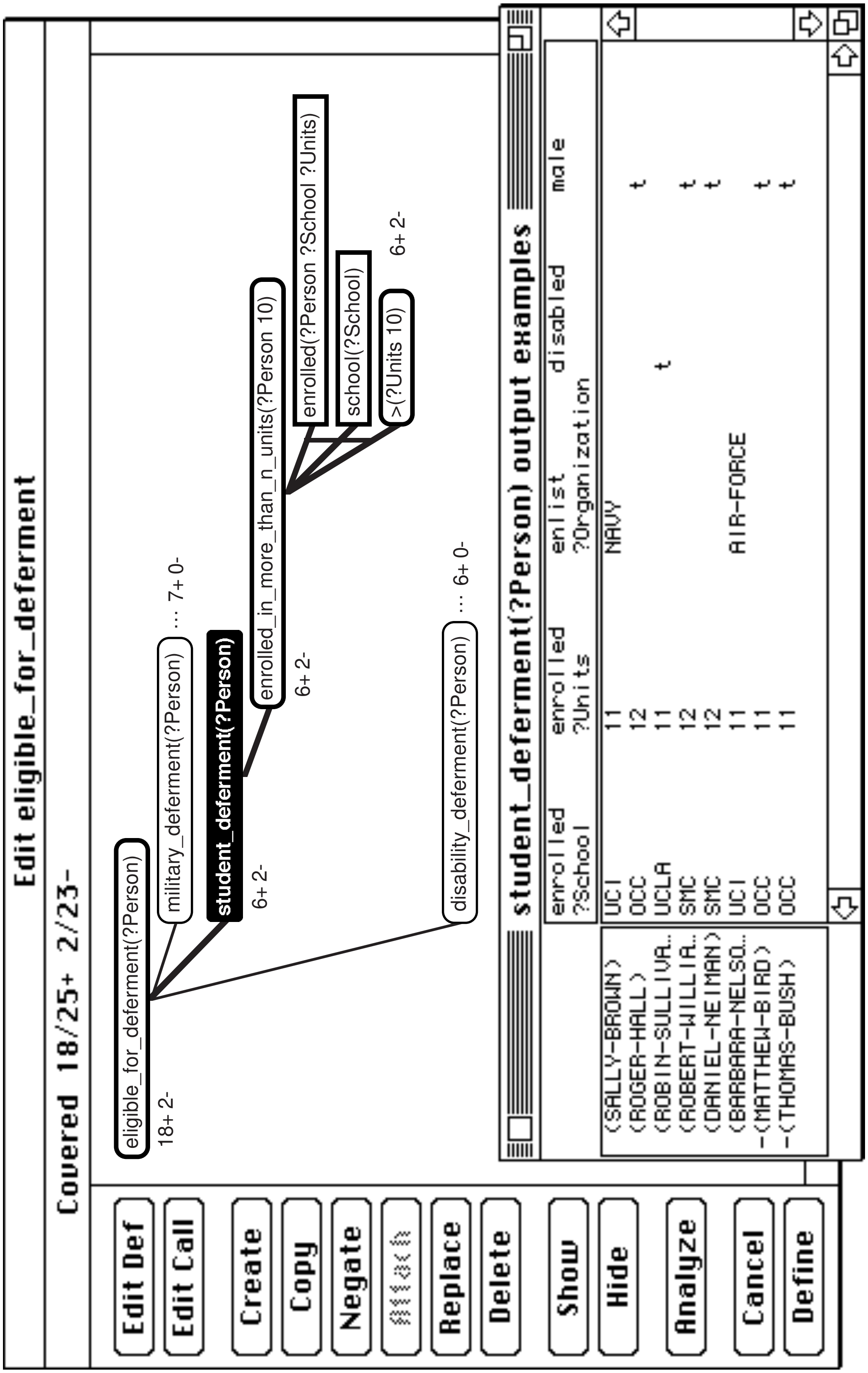

Figure 1. Finding the examples that satisfy the conditions of a rule.

of the students who are required to pay back their loans are enrolled in more than 11 units. This display gives a knowledge engineer familiar with the domain the information needed to revise the definition, by increasing the number of units required for a student deferment to more than 11.

In the remainder of this paper, we first describe the learning system, FOCL. Next, we discuss FOCL-1-2-3 and the extensions that have increased its utility as a knowledge acquisition tool. Finally, we discuss limitations of the system, outline the direction of our future work and draw preliminary conclusions about the approach we have taken.

2.0 FOCL

FOCL is a multistrategy relational learning program based on the inductive learning algorithm FOIL (Quinlan, 1990). In this section, we give a brief overview of FOCL for those unfamiliar with the system. For a detailed description of the FOCL algorithm and a complexity analysis see (Pazzani & Kibler, 1992).

FOCL uses a separate-and-conquer approach guided by an information-based heuristic to produce a set of clauses such that all the positive examples are covered by at least one clause and no negative example is covered by any clause. Even though FOCL employs a number of different learning strategies, it uses a single information-based evaluation function to guide learning within each component and to control the interaction between components. FOCL was designed to learn Horn-clause descriptions by combining analytical and empirical learning. As a consequence of this combination of learning methods, FOCL can utilize incomplete and incorrect domain theories (Mitchell, Keller, & Kedar-Cabelli, 1986; Rajamoney, & DeJong, 1987). FOCL uses the rule base of an expert system as the domain theory required by explanation-based learning and processes a collection of examples. FOCL extracts the portions of the domain theory that correctly classify examples, patches the portions of the domain theory that are useful but incorrect and discards the portions which have no information gain.

FOCL can be viewed as having two operators: **add-clause** which starts a new clause with the body true, and **add-literals** which adds literals to the end of the current clause body. FOCL performs the **add-literals** operator until no negative examples are covered by the clause. When a clause is formed, the positive examples covered by that clause are removed and FOCL performs the **add-clause** operator on the remaining examples, adding new clauses until all positive examples are covered by some clause. FOCL, like FOIL, uses an information-based metric to evaluate when to add a literal to a clause under construction. However, unlike FOIL, FOCL can use either an empirical (i.e., inductive) component or an analytical (i.e., explanation-based) component to add literals to a clause. The result of this combination of methods is that the system can make use of domain knowledge when available to constrain the search for a concept definition. However, when the domain knowledge is incomplete or incorrect, the literals proposed by the analytical component will have less information gain than the literals proposed by the empirical component and new literals will be added by empirical methods.

2.1 Empirical Learning Components

The empirical learning components make use of three kinds of semantic knowledge to constrain search: argument type information (whether a specific argument must be of a specific sort), argument mode information (whether a specific argument must be bound or unbound) and argument constraint information (e.g., whether the predicate is commutative). In addition to checking all possible bindings of arguments to variables, it is possible for FOCL to consider argument bindings which contain constants.

2.1.1 Induction

Within FOCL, inductive methods for adding literals compute the information gain of every consistent binding of every relation and select the relation and argument binding with the highest gain. There are three variants of the induction component. The extensional induction component which is identical to FOIL performs induction over the relations defined as sets of positive and negative instances. The built-in induction component performs induction over relations defined as primitive lisp functions such as < and `equal`. The intensional induction component performs induction over the relations defined as rules. If an intensional relation has the maximum information gain, it is operationalized in the same manner that the target concept is operationalized by the analytical-component. This process is discussed in Section 2.2.

2.1.2 Cliché Instantiation

Clichés (Silverstein & Pazzani, 1991) are templates that specify relation and variablization restrictions on the generation of conjunctions of literals. Clichés differ from the rule models of MOBAL and the second-order schemata of CLINT in that they specify restrictions on both the relations which can be conjoined and on the arguments bindings of those relations. The cliché instantiation mechanism allows the empirical component a limited form of multiple step look ahead which aids FOCL's normal hill-climbing approach. The restrictions imposed by the cliché constrain the exponential explosion which would result from trying all pairs or triples of literals.

2.1.3 Determinate Addition

Muggleton & Feng (1990) introduced the notion of using determinacy and Quinlan (1991) incorporated a similar idea into FOIL. FOCL's implementation of determinacy is similar to FOIL's except that FOCL uses determinacy templates to create determinate literals. A determinacy template for a relation specifies a mode restriction and an associated determinacy. For instance the determinacy templates for the relation

father(?father ?child) = { (mike karen) (mike lynn) (jim lisa) (jim cynthia) (jim mike) (bob brit) }

would be (((:- :+) 1) ((:+ :-) 3)). This indicates that if ?child is specified then ?father can be determined uniquely, while if ?father is specified then in this data set there are at most three possible values for ?child. When all other learning components fail to produce a literal or conjunction with positive information gain, the determinate addition component adds all determinate literals of a specified determinacy to the current clause, then learning continues from this new state. The assumption is that the determinate literals will provide some bindings that further learning can take advantage of.

2.1.4 Simplification (Clause Generalization)

The simplification process can be invoked when FOCL has finished learning a clause, or when the analytical component has found a useful conjunction of literals. The simplification process deletes literals from the clause or conjunction that do not contribute to the definition. The deleted literals either unnecessarily exclude positive examples or exclude no negative examples. Simplification is helpful in generalizing overly specific domain knowledge, and in overcoming the effects of greedy search (Quinlan, 1990).

2.2 Analytical Learning Components

Like all explanation-based learning systems, FOCL's analytical component requires a target concept, (i.e., a non-operational definition of the concept to be acquired), a domain theory (i.e., a set of rules, such as the rule base of the expert system, that relates the non-operational definition to operational predicates), a set of operational predicates (i.e., features that the examples are expressed in terms of), and a set of classified training examples. FOCL computes the information gain of the target concept. If the target concept has positive information gain, then FOCL extracts the portion of the target concept that gives correct classification by an operationalization process. Otherwise, FOCL employs an empirical component. This strategy insures that FOCL uses as much of the user provided domain theory as possible, but disregards those parts of the domain theory that misclassify large numbers of examples.

In FOCL, the operationalization process differs from that of EBL in that it attempts to find a proof that is true of many positive examples and few negative examples rather than a proof of a single example. The operationalization process descends the path of maximum information gain from the target concept through the domain theory until it finds a conjunction of extensionally defined literals. These literals are then added to the clause under construction where they may be specialized by the empirical component (if the proof explains some negative examples.) This process is illustrated graphically in Figure 2.

3.0 FOCL-1-2-3

FOCL-1-2-3 is a knowledge acquisition environment based on FOCL that incorporates a graphic user interface that allows: dynamic visualization of the learning process, selection of alternate hypothesis to direct low level learning, and manipulation of both facts and rules. In addition FOCL-1-2-3 includes mechanisms to define examples in terms of relations and analyzing rule coverage. In keeping with the balanced cooperative approach it is possible to invoke the learning algorithm on any sub-problem in the knowledge base. The user specifies a learning task by selecting a set of examples and possibly a target concept. Alternatively, the user can enter definitions for new relations or modify existing relations at any time.

3.1 Visualizing the Learning Process

We believe that dynamic visualization of the learning process will prove helpful for anyone using FOCL-1-2-3. Instead of developing an abstract mental model of FOCL's learning process, there is an explicit graphic representation that the user can relate to. This explicit representation has made the learning algorithm much easier to observe and explain. We believe that integrating this kind of visualization technique into a knowledge acquisition tool can facilitate the exchange of information between the knowledge engineer and the domain expert.

Figure 2 is a snap shot of the operationalization process described in section 2.2. In the figure, there are 12 positive and 23 negative examples of no_payment_due remaining. There are two alternative ways that no_payment_due could be true (continuously_enrolled or eligible_for_deferment). The first is true of 9 positive examples and 13 negative examples, while the second is true of 8 positive and 9 negative. Because the second has higher information gain, FOCL recursively operationalizes it and finds that peace_corp_deferment and student_deferment both are true of 2 positive and no negative examples. The first is selected randomly for further operationalization and eventually the operationalization process returns the conjunction enlist(?0 ?1) and peace_corps(?1) which represents the rule that a person enlisted in the peace corps is not required to make student loan payments.

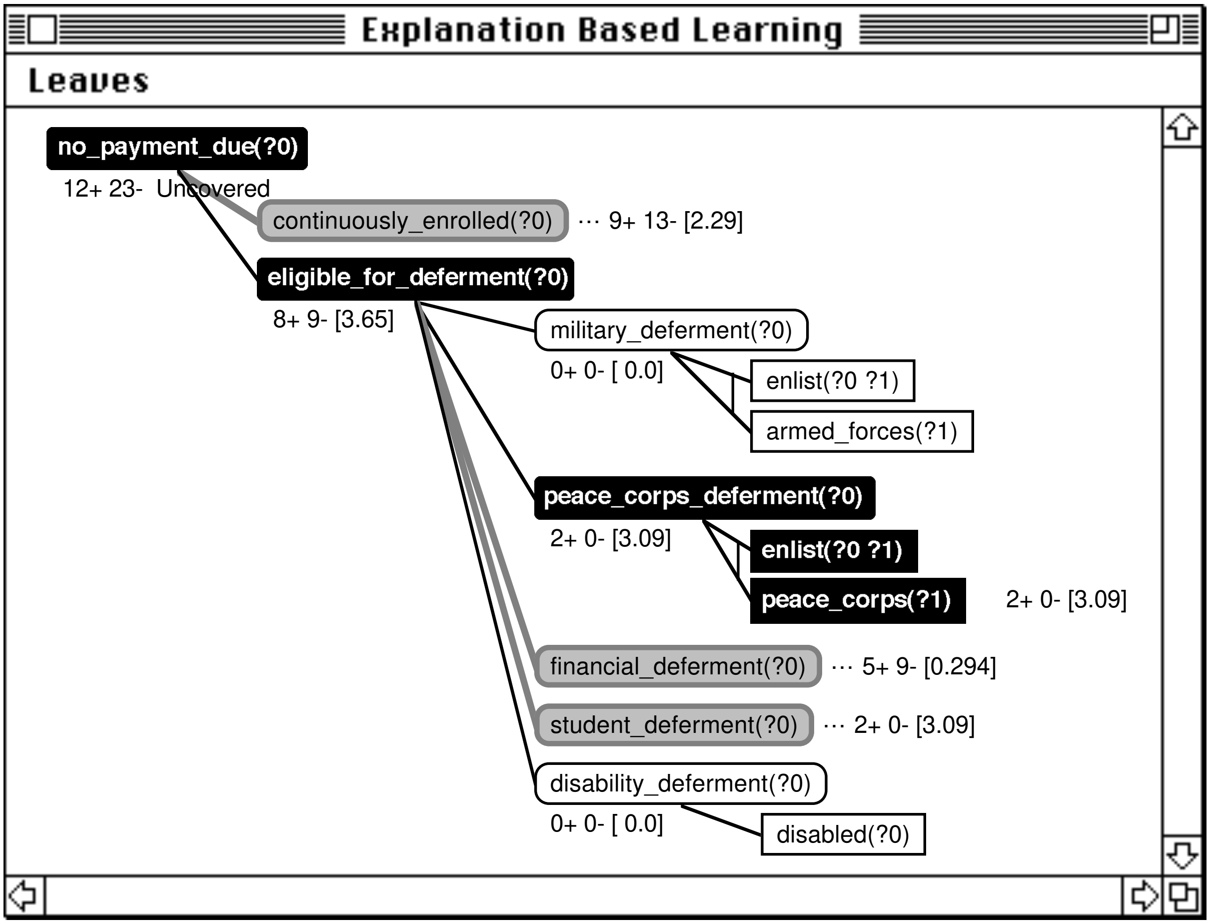

Figure 2. FOCL's Operationalizer

3.2 Examples and Relations

A subtle problem we had to address is that when dealing with a relational learning system the notion of an example is fuzzy. From the learning algorithm's perspective the universe is composed of relations. However, most people feel more comfortable dealing with examples. In attribute-value learning systems this distinction does not arise. Objects have a predetermined set of features. However in a relational learning system, objects as such do not exist, rather they are the values of a relation and are connected by other relations to other values. For instance, in an attribute value learning system (clifford-brown 12 UCI T) might be presented to the system as an example. While in a relational learning system the same information would almost certainly be distributed over a number of relations such as (name S14 clifford-brown) (enrolled S14 UCI 12) (male S14).

The primary feature that distinguishes relational learners from attribute value learners is their ability to access data which is not explicitly contained in an example. For instance, in the student loan example, the fact than a school is a four year college is a condition that can be incorporated into a learned rule, while this fact is not about an individual student. A relational language provides additional expressiveness to the learned rules but also complicates the task of displaying an example, or even deciding what an example is. Should (four-year-university UCI) be presented to the user when information about student S14 is requested? Our solution is to supply FOCL-1-2-3 with example templates that represent user defined ways of viewing information from the database. Example templates tell the system what related information to present when displaying an instance of a specific relation. Example templates can be created manually or a default view is created automatically.

If the user asks to see some examples of a particular predicate the system retrieves the associated template, collects the related data and presents the user with the desired information. For instance in Figure 3, the system is displaying the examples of no_payment_due(?Person) that satisfy the literal enlist(?Person ?Organization). Although, the tuples that satisfy enlist(?Person ?Organization) are of the form (:person :organization) the example template for no_payment_due specifies that addition information including length of absence from school and employment status should be presented.

3.3 Manipulating Examples and Relations

In order to allow the user to deal with large numbers of examples in a coherent fashion, we have incorporated a spread sheet style interface into FOCL-1-2-3. The interface allows the user to inspect and manipulate a set of examples in a fashion which is analogous to a spread sheet.

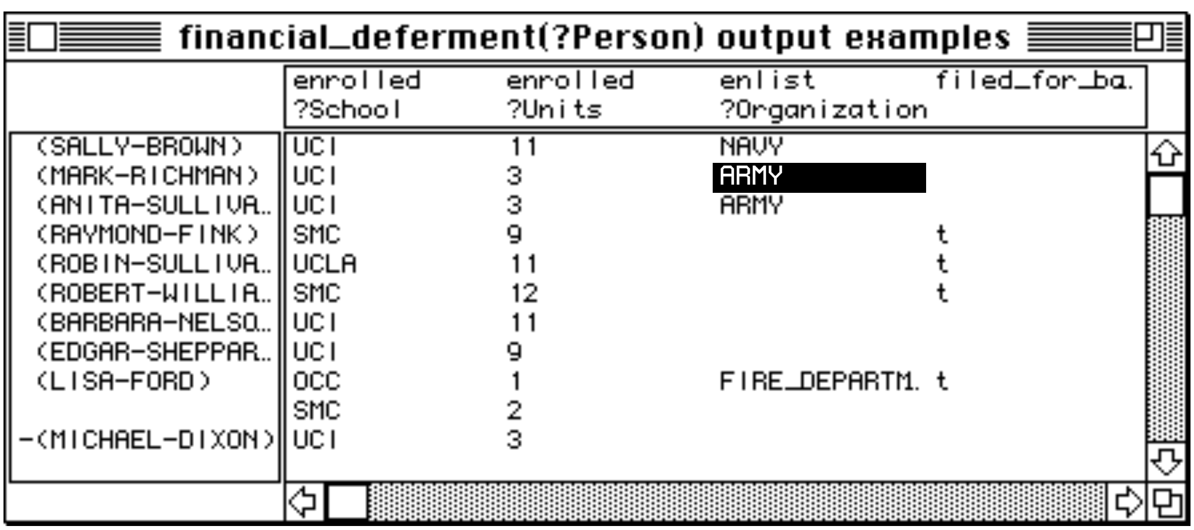

Figure 3. Spread Sheet Style Fact Interface

Figure 3 is a snapshot of this interface. Example labels are displayed in the box to the right of the window. Relation labels are displayed in the box at the top of the window. The first line of a relation label is the relation name and the second line is the relation argument. All of the information associated with an example appears in the same row as the example label. For instance the school Mark Richman is enrolled in can be found in row `MARK-RICHMAN` and column `enrolled ?School`. If there are multiple values for a given relation they appear as multiple entries. For instance, Lisa Ford is enrolled both at OCC and SMC. For relations arity two and higher relations the values of all none example label bindings are displayed in the table. Arity one relations are handled specially. Instead of displaying the argument bindings, the truth value of the relation is displayed. For example, filed_for_bankruptcy (lisa-ford) is a fact in the knowledge base and therefore the corresponding cell contains a `t`.

Using the spread sheet style interface, a value within a relation can be changed by selecting the associated cell in the display and entering the new value. All assertions associated with a given example can be retracted from the knowledge base by deleting the cell associated with the example. Similarly, relations can be deleted from the knowledge base by deleting the cell associated with the relation. In a future implementation, it will also be possible to add new examples and new relations to the knowledge base in a similar fashion. Once the modifications are complete, the changes are propagated throughout the system and the change in the coverage of any rule will be reflected.

In addition to the interface for manipulating examples and extensionally defined relations, we have created a graphic rule editor for FOCL-1-2-3. The rule editor displays a graphic representation of an intensionally defined relation. Using the rule editor a knowledge engineer can add literals to a clause, delete literals from a clause, reorder literals within a clause, add clauses to a rule, delete clauses from a rule, and change the rule's argument specifications. An innovative feature of the rule editor is that it retains the examples from the context in which it was invoked. Then the effect of any modification to the rule can be analyzed in terms of the coverage of these examples. For instance, Figure 4 shows summary information next to each clause. The specific examples can also be displayed (as in Figure 1). In cases where multiple uses of the same rule occur, the examples presented to each of these occurrence are collected and input to the editor. Since the editor environment is distinct from the knowledge base, the expert can evaluate alternate rule definitions without affecting the entire knowledge base. As with the example interface, once the modifications are complete the changes are propagated throughout the system.

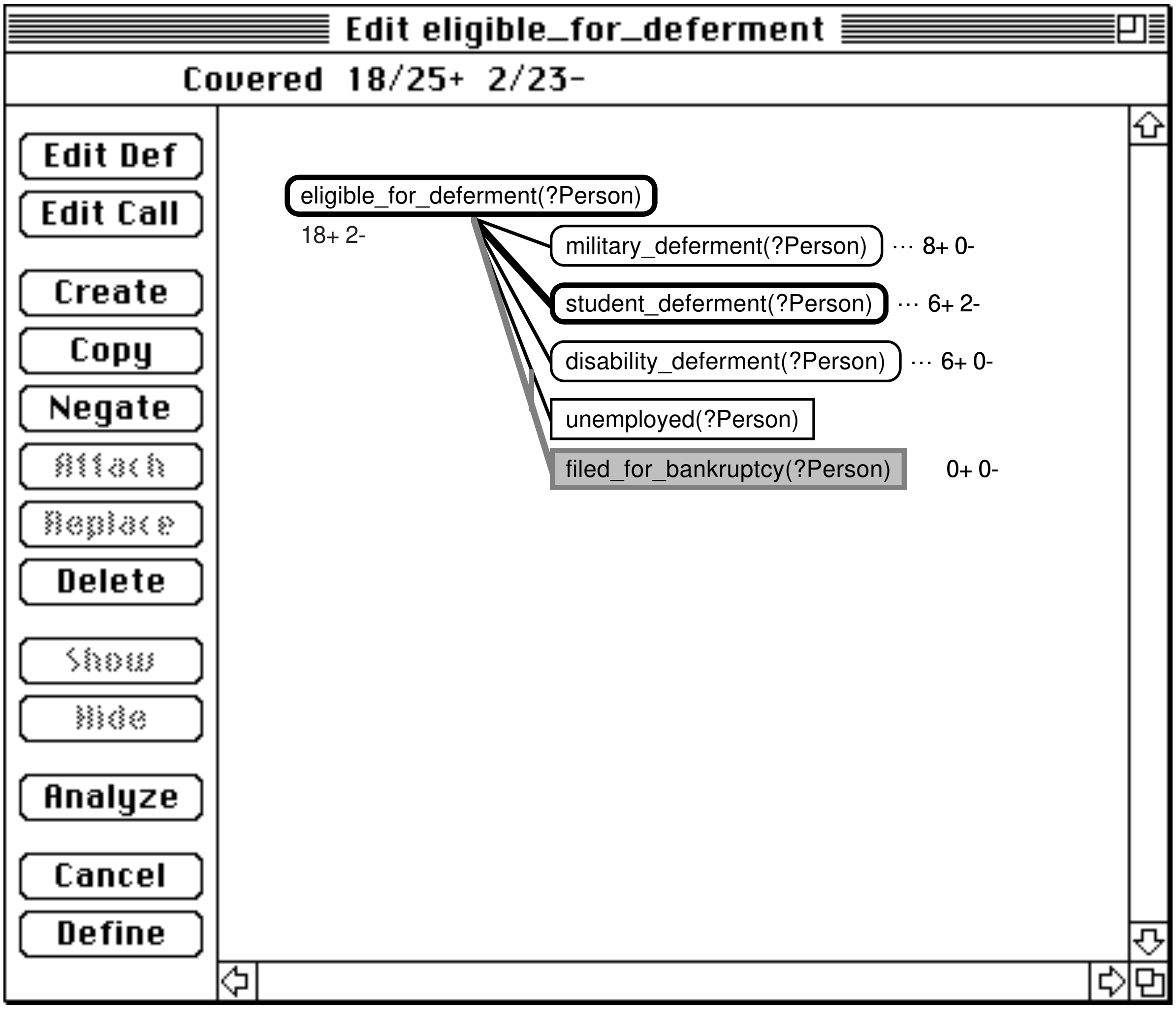

Figure 4. Graphic Rule Editor

3.4 Low Level User Interaction

We have extended the interaction between the user and the learning algorithm beyond the level of the balanced cooperative approach by allowing the user to interact with the learning algorithm at every level. Parameter setting, problem specification, and example selection is possible as it is with all learning algorithms. Algorithm directed interaction with the user (ala CLINT) is possible. Interactive modification of knowledge at anytime by user (ala MOBAL) is possible. Unlike other learning systems, FOCL-1-2-3 allows low level selection of the learning path, through a simple literal selection process. If so desired, expert interaction can be incorporated directly into the low level control structure of the learning algorithm. In this process, the learning algorithm explores the space of possible literals to add, collecting statistics on the examples covered by various alternatives, but the selection of which literal to add to the clause under construction is performed by the knowledge engineer.

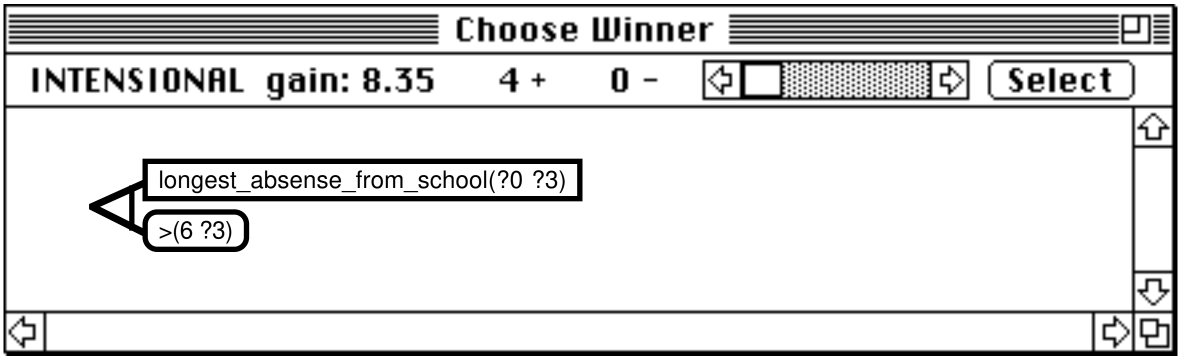

Figure 5. FOCL allowing expert to select from candidate extensions

The learning algorithm finds a specified number of the best candidates for extending the current clause from all the literals and conjunctions tested. The expert can then select the candidate to continue the learning process. The candidates are ordered according to decreasing information gain. In addition to the candidate, the candidate's gain, and the learning component that generated the candidate; the expert can see the clause being extended and the examples covered. We believe that this low level control, combined with the easy access to relevant information provides a useful extension to completely automated learning. Although information gain is very useful as a guide, there are times when an expert can provide insight, particularly when there is a semantic preference for some literal combination that cannot easily be captured in a selection function used by an automated learning system.

3.5 Analyze Coverage

To allow a knowledge engineer to explore the implications of changes to the knowledge base and to locate problematic portions of the knowledge base, we have created a tool that analyzes the coverage of the target concept. In Figure 6, the coverage analysis has been performed on the target concept no_payment_due using examples from the extensionally defined relation no_payment_due_fact. This tool displays the number of positive and negative examples that satisfy each clause and highlights literals that cover negative examples. After analyzing the coverage it is possible to display the set of examples that each literal covers. Reasoning abstractly about rules can be difficult, examples help to ground this process. In addition to maintaining the examples that each rule covers during analysis, FOCL also maintains the coverage generated at learning time and allows the expert to access this information.

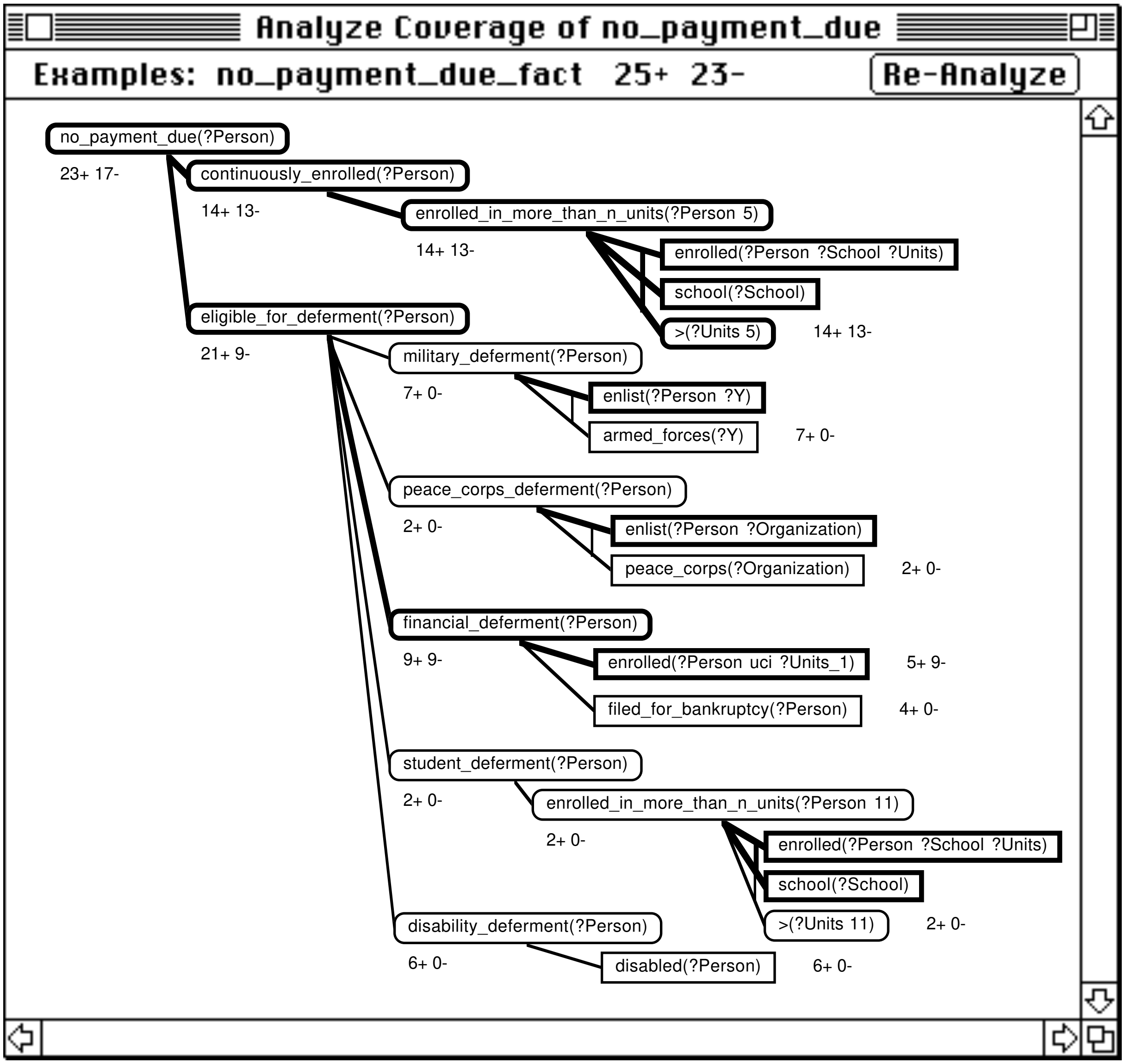

Figure 6. Analysis of Goal Concept Coverage

3.5 Knowledge Revision

Previously we reported on a prototype knowledge acquisition tool KR-FOCL (Pazzani & Brunk, 1990) that partially automates the task of identifying rules responsible for errors in expert systems. KR, the knowledge revision component, was designed to use three heuristics and a derivation of the learned descriptions generated by FOCL to suggest revisions to the knowledge base. KR has been integrated into FOCL-1-2-3. The interface to KR is a specialized version of the rule editor. After learning KR can be invoked.

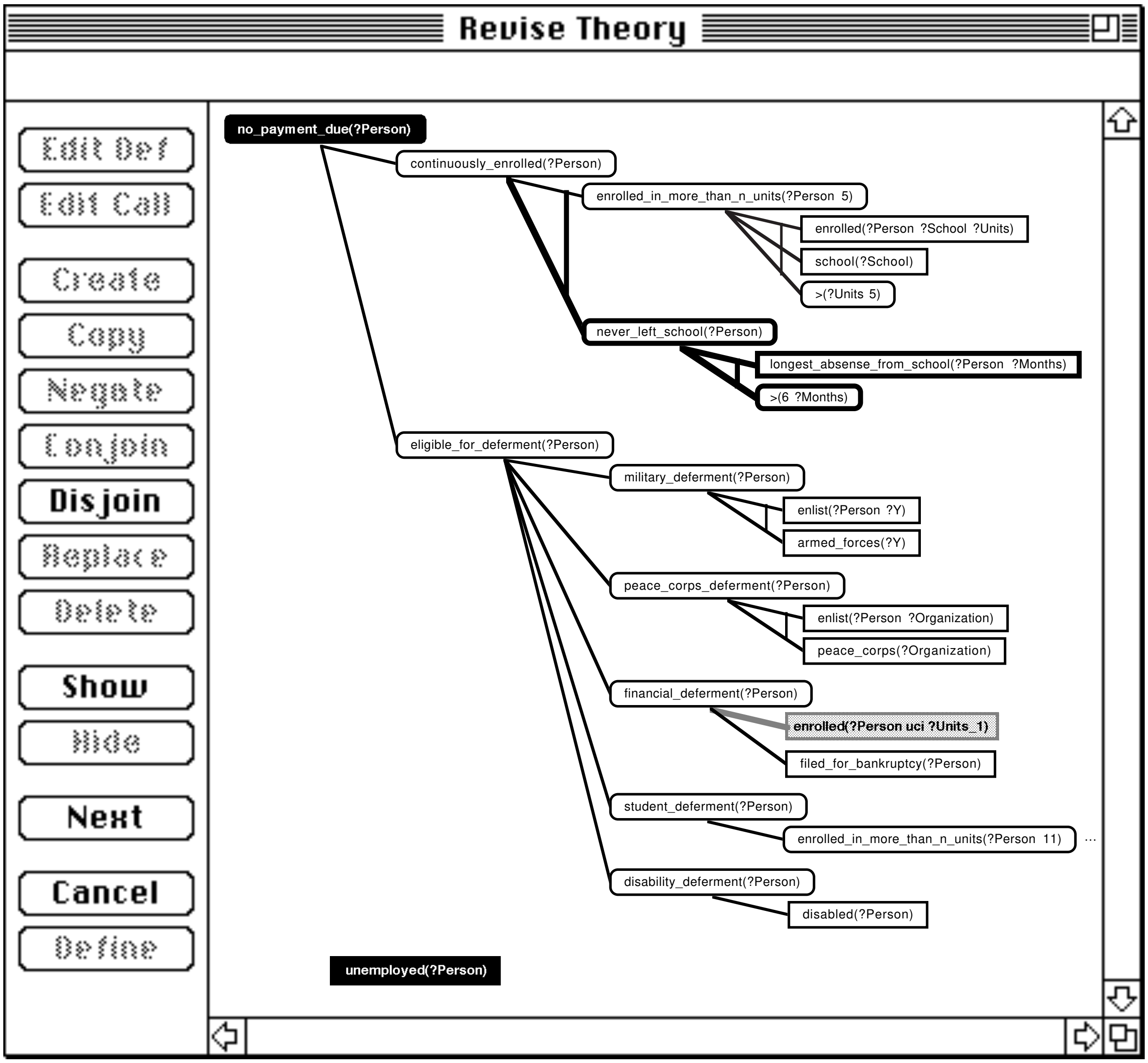

Figure 7. Knowledge Revision using KR

First, KR collects all the clauses from the learned description that were learned via a combination of empirical and analytical techniques. It then clusters those clauses that were learned by operationalizing the same path through the domain theory extracting the empirical literals from the clauses in each cluster. Sequentially, the operationalized path and empirical literals from each cluster are presented to the user as disjoint graphs. Empirical literals from the same clause appear as conjunctions, while literals from different clauses appear in separate disjuncts. The default attachment point from the learned description is indicated. The user is able to accept the indicated attachment, select an alternate conjunction to which the empirical literals will be conjoined, or move on to the next revision suggestion. In Figure 7, the user has conjoined the induced clause never_left_school(?Person) to the operationalized literal enrolled_in_more_than_n_units(?Person).

Next, KR collects all the clauses from the learned description that were learned via empirical methods alone. Again the user is presented with two disjoint graphs only this time the user is asked to select an attachment point for which the empirical literals will form a new disjunct. Figure 7 illustrates this. During learning the induced clause unemployed(?0) was learned as a disjunct for no_payment_due(?0). The knowledge engineer is presented with this default attachment point and given the option selecting another.

Finally, KR suggests deleting literals and clauses that were not needed to obtain correct classification. These include literals that were removed during a simplification process and portions of the domain theory that were never operationalized.

FOCL-1-2-3 addresses some of the limitations of KR-FOCL. KR-FOCL did not explain why it recommended certain revisions. FOCL-1-2-3 maintains a more complete derivation which includes the examples that satisfy each literal. Combined with a description of the revision heuristics these examples provide an explanation of the revision recommendation. Instead of serially asking verification questions about a large number of possibilities as KR-FOCL did. FOCL-1-2-3 presents the expert with a graphic editor and a suggested revision. The expert can choose to accept the revision as is, propose a different revision, or skip to the next suggestion. Additionally, we have begun to add capabilities similar to programs such as SEEK (Politicks & Weiss, 1984) that can quickly assess the impact of revision. It is no longer necessary to re-run FOCL or re-analyze the entire target concept to determine the impact of a change. Re-analysis is performed automatically and limited to the affected portions of the knowledge base.

3.6 Other Tools

Most of FOCL-1-2-3 is designed to help the user identifying logical errors in a rule base. However, it also checks the syntactic structure of the rule base. Currently, four types of errors are detected: using any predicates that are not defined, using the wrong number of arguments in a predicate, using arguments that are not consistent with the type declaration and making sure each variable in a clause is used more than once. If any of these errors are detected, FOCL-1-2-3 collects all warnings about a rule, and opens up an editor window that allows the knowledge engineer to see the warning messages and to correct the rule definition. Figure 8 illustrates one such example.

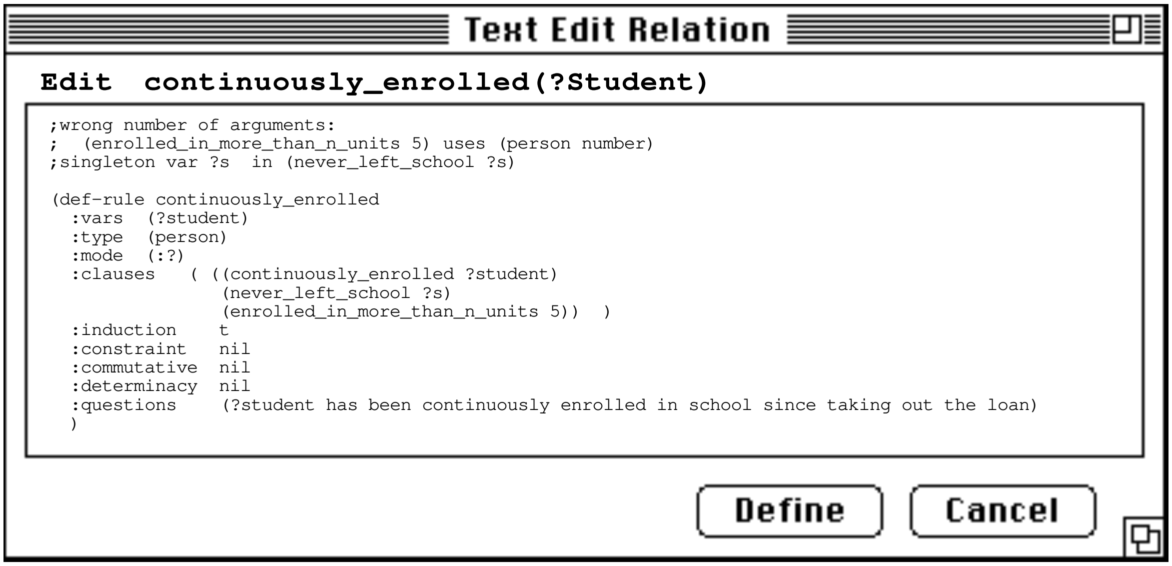

Figure 8. Editing a rule with stylistic warning

4.0 Limitations and Future Work

FOCL-1-2-3 is limited by the underlying machine learning technology; it is designed to work only with rule-based expert systems that perform classification tasks. If the problem cannot easily be expressed using Horn-clauses or if the rules require certainty factors or if the expert system relies on a complex conflict resolution strategy to determine the applicability of rules, then FOCL-1-2-3 is not an appropriate tool to use. If FOCL cannot solve the learning problem by using hill-climbing search, then FOCL-1-2-3 reduces to a fancy editing environment. We are currently investigating extensions to the analytical component that will allow FOCL to learn from a larger class of erroneous domain theories. One approach we are investigating is the use of a mechanism which allows disjunctive frontiers to be operationalized in a single pass rather than requiring the operationalization of conjunctive paths in isolation. We are also investigating the integrating abductive techniques to localize blame and direct learning of intermediate concepts.

As a theory revision tool FOCL-1-2-3 is limited in that it does not perform revisions autonomously, instead it presents revision suggestions to the user who performs the actual modification. However, the new operationalization strategy mentioned above offers a natural mechanism for performing revision at learning time. We plan to investigate the utility of this approach compared to the current approach of revision only after learning a complete concept definition.

Another of FOCL's limitations is that learning occurs at a very syntactic level. Currently, the only semantic knowledge that FOCL utilizes comes in the form of a limited ontology of relations used by the cliché instantiation mechanism and a rudimentary type system. By extending the ontology and integrating it into the other learning components, it would be possible using a relevance metric to decrease the space that FOCL searches. Similarly, by enhancing the sort system to handle hierarchical typing, the search space can be reduced thereby increasing the effectiveness of the learning components.

FOCL-1-2-3 does not address the problem of knowledge integration. Since the system does not handle structured objects, knowledge integration usually devolves to relation modification. However, sophisticated knowledge integration techniques like those found in KI (Bareiss, Porter & Murray, 1989) can be incorporated as we begin to deal with more complex representation languages and structured objects.

Currently, we have only tested FOCL-1-2-3 on a small number of knowledge bases: the student loan domain, a trade negotiations domain, and a track identification domain. FOCL-1-2-3's interface is Macintosh dependent, as a result the size of the domains are limited. However, the underlying learning algorithm which is machine independent has been tested on larger problems and appears quite robust. We are in the process of expanding FOCL-1-2-3's repertoire.

We are developing a protocol to evaluate the effectiveness of the extensions in FOCL-1-2-3. We plan to conduct two experiments in the context of an upper division undergraduate class on expert systems. The first experiment will assess the new learning-revision mechanism's ability to perform autonomous theory revision and compare it to the students' ability. The second experiment will evaluate the utility of the learning-revision mechanism in the knowledge acquisition process. Students will construct a knowledge base using FOCL-1-2-3 with and without the learning-revision component. The creation time, accuracy and generalizability of each knowledge base will be compared. Systematically evaluating the utility of each of the knowledge acquisition extensions incorporated into FOCL-1-2-3 is important, but the time required to do so is prohibitive. Thus, we are focusing on what we feel to be the most important component. However, we intend to survey the class to determine their feelings about the utility of various components.

5.0 Conclusions

In a large system, it is difficult for a user to anticipate the impact that a change made to the knowledge base will have on the coverage and correctness of the classification of examples in a database. We have attempted to address this issue by developing graphical methods to summarize the effect of changes. FOCL-1-2-3 is a further step in the integration of knowledge acquisition and machine learning techniques. It offers an open learning algorithm and a balanced cooperative approach to knowledge acquisition while incorporating a number of novel features including:

- a dynamic graphic representation of the learning process,
- a rule editor that can assess changes to a rule in terms of local example coverage,
- a spread sheet like interface for manipulating examples and facts,
- a mechanism for quickly assessing the affect of modifications to the knowledge base, and
- a technique for maintaining a connection between rules and the examples they cover.

This latter technique is useful for providing a context in which the expert can reason about rules. In the future these examples will provide the basis for automation of the revision process. We envision a revision component that interacts with the expert in a fashion similar to the learning component. The revision component will analyze a myriad of possible revisions and cull from among them a specified number of the best that will be suggested to the knowledge engineer.

One feature that distinguishes FOCL from many other learning systems is its ability to perform recurrent learning. FOCL can utilize the descriptions it learns as background knowledge and is therefore capable of assisting in complex theory formation tasks. As a trivial example, assume that FOCL had not been provided with an intensional definition of continuously_enrolled, but had been provided with a set of examples of students who were continuously_enrolled. FOCL could be used to learn an intensional definition of continuously_enrolled. This rule could then be used as background knowledge when learning a more complex concept like no_payment_due or eligible_for_scholarship. Although, this type of theory formation process is difficult to control automatically, it is simple to direct manually, and it is a natural nexus between knowledge acquisition and machine learning. We believe that FOCL-1-2-3 provides an excellent environment for exactly this type of activity.

References

Bareiss, R., Porter, B., & Murray, K. (1989). Supporting Start-to-Finish Development of Knowledge Bases. *Machine Learning, 4*, 259–283.

Davis, R. T. (1978). Model-directed learning of production rules. In D. Waterman & F. Hayes-Roth (Eds.), *Pattern-directed inference systems*. New York: Academic Press.

De Raedt, L., & Bruynooghe, M. (1989). Towards friendly concept-learners. *Proceedings of the Elenenth International Joint Conference on Artificial Intellegence.* Morgan Kaufmann.

De Raedt, L., & Bruynooghe, M. (1992). Interactive Concept-Learning and Constructive Induction by Analogy. *Machine Learning, 8*, 107–150.

Marcus, S. McDermott, J. & Wang, T. (1985). Knowledge acquisition for constructive systems. *Proceedings of the Ninth International Joint Conference on Artificial Intelligence*. Los Angeles, CA: Morgan Kaufmann.

Mitchell, T., Keller, R., & Kedar-Cabelli, S. (1986). Explanation-based learning: A unifying view. *Machine Learning*, *1*, 47–80.

Muggleton, S. & Feng, C. (1990) Efficient induction of logic programs. *Proceedings of the First Conference on Algorithmic Learning Theory,* Tokyo, Japan. Ohmsha.

Morik, K. (1991). Balanced Cooperateive Modeling. *Proceedings of the FirstInternationWorkshop on Multistrategy Learning.* 65–80. Harpers Ferry.

Ourston, D., & Mooney, R. (1990). Changing the rules: A comprehensive approach to theory refinement. *Proceedings of the Eighth National Conference on Artificial Intelligence* . 815–820. Boston, MA: Morgan Kaufmann.

Pazzani, M., & Brunk, C. (1990). Detecting and correcting errors in rule-based expert systems: an integration of empirical and explanation-based learning. *Proceedings of the Fifth Knowlegde Acquisition for Knowledge-Based Systems Workshop,* Banff, Alberta: SRDG Publications.

Pazzani, M., Brunk, C., & Silverstein, G. (1991). A knowledge-intensive approach to learning relational concepts. *Proceedings of the Eight International Workshop on Machine Leanring*. 432–436. Evanston, Il: Morgan Kaufmann.

Pazzani, M., & Kibler, D. (1992). *The utility of knowledge in inductive learning* . *Machine Learning, 9*, 57–94.

Politakis, P & Weiss, S. (1984). Using empirical analysis to refine expert system knowledge Bases. *Artificial Intelligence, 22*, 23-48.

Quinlan, J. R. (1986). Induction of decision trees. *Machine Learning, 1,* 81-106

Quinlan, J. R. (1990). Learning logical definitions from relations. *Machine Learning, 5*, 239–266.

Quinlan, J. R. (1991). Determinate Literals in Inductive Logic Programming. *Proceedings of the Eight International Workshop on Machine Leanring*. 442–446.Evanston, Il: Morgan Kaufmann.

Rajamoney, S., & DeJong, G. (1987). The classification, detection and handling of imperfect theory problems. *Proceedings of the Tenth International Joint Conference on Artificial Intelligence* . 205-210. Milan, Italy: Morgan Kaufmann.

Richards, B. & Mooney, R. (1991). First-Order Theory Revision. *Proceedings of the Eight International Workshop on Machine Leanring.* 447–451. Evanston, Il: Morgan Kaufmann.

Shavlik, J., & Towell, G. (1989). Combining explanation-based learning and artificial neural networks. *Proceedings of the Sixth International Workshop on Machine Learning.* 90–93. Ithaca, NY: Morgan Kaufmann.

Silverstein, G., & Pazzani, M. (1991). Relation clichés: Constraining constructive induction during relational learning. *Proceedings of the Eight International Workshop on Machine Learning.* 203–207. Evanston, Il: Morgan Kaufmann.

Wogulis, J. (1991). Revising Relation Domain Theories. *Proceedings of the Eight International Workshop on Machine Learning.* 462–466. Evanston, Il: Morgan Kaufmann.