AUTOMATED REVISION OF CLIPS RULE-BASES

Patrick M. Murphy, pmurphy@ics.uci.edu
Michael J. Pazzani, pazzani@ics.uci.edu

Department of Information & Computer Science
University of California, Irvine, CA 92717

Abstract

This paper describes CLIPS-R, a theory revision system for the revision of CLIPS rule-bases. CLIPS-R may be used for a variety of knowledge-base revision tasks, such as refining a prototype system, adapting an existing system to slightly different operating conditions, or improving an operational system that makes occasional errors. We present a description of how CLIPS-R revises rule-bases, and an evaluation of the system on three rule-bases.

INTRODUCTION

Considerable progress has been made in the last few years in the subfield of machine learning known as theory revision, e.g. [1,2,3]. The general goal of this area is to create learning models that can automatically update the knowledge base of a system to be more accurate on a set of test cases. Unfortunately, this progress has not yet been put into common practice. An important reason for the absence of technology transition is that only a restricted form of knowledge bases have been addressed. In particular, only the revision of logical knowledge bases that perform classification tasks with backward chaining rules [4] has been explored. However, nearly all deployed knowledge-based systems make use of forward-chaining production rules with side effects. For example, two of the knowledge-based systems reported on at the 1993 Innovative Applications of Artificial Intelligence use CLIPS. The remainder of the knowledge-based systems use ART, a commercial expert system that has many of the same features as CLIPS.

There are a variety of practical reasons why the production rule formalism is preferred to the logical rule formalism in deployed expert systems. First, production rules are suitable for a variety of reasoning tasks, such as planning, design and scheduling in addition to classification tasks that are addressed by logical rules. Second, most deployed knowledge-based systems must perform a variety of computational activities such as interacting with external databases or printing reports in addition to the "reasoning" tasks. The production system formalism allows such procedural tasks to be easily combined with the reasoning tasks. Third, the production rule systems tend to be computationally more efficient. The production systems allow the knowledge engineer

to have more influence over the flow of control in the systems allowing the performance to be fine tuned. Whereas in a logical system, the rules indicate what inferences are valid, in a production system, the rules indicate both which inferences are valid and which inferences should be made at a particular point.

The revision of CLIPS rule-bases presents a number of challenging problems that have not been addressed in previous research on theory revision. In particular, rules can retract facts from working memory, display information, and request user input. New opportunities to take advantage of additional sources of information also accompany these new problems. For example, a user might provide information that a certain item that was displayed should not have been, or that information is displayed in the wrong order.

In the remainder of this paper, we give an overview description of CLIPS-R, a system for revising CLIPS rule-bases and an evaluation of CLIPS-R on three rule-bases.

DESCRIPTION

CLIPS-R has an iterative refinement control structure (see Figure 1). The system takes as input a rule-base and a set of instances that define constraints on the correct execution of the rules in the rule-base. While there are unsatisfied constraints CLIPS-R heuristically identifies a subset of similar instances as problem instances (instances with many unsatisfied constraints). Using the problem instances, a set of potential repairs to the rule-base are heuristically identified. Each of these repairs is used to temporarily modify the rule-base, with each modified rule-base evaluated over all instances. The repair that improves the rule-base most is used to permanently modify the rule-base. If no repair can increase the evaluation of the rule-base, than the addition of a new rule through rule induction is attempted. If rule induction cannot generate a rule that can improve the evaluation of the rule-base, the revision process halts and the latest rule-base is returned. The process of modification and rule induction continues until all constraints are satisfied or until no progress is made.

Instances

Each instance has two components: initial state information and constraints on the execution of the rule-base given the initial state. The initial state information consists of a set of initial facts to be loaded into the fact-list before execution of the rule-base, and a set of bindings that relate input function calls to their return values. From the automobile diagnosis rule-base, the function *ask-question* with argument "What is the surface state of the points?" may return "burned" for one instance and "contaminated" for another instance. The set of constraints on the execution of the rule-base includes constraints on the contents of the final fact-list (the final fact-list is the fact-list when execution of the rule-base halts), and constraints on the ordering of observable actions such as displaying data or asking the user questions.

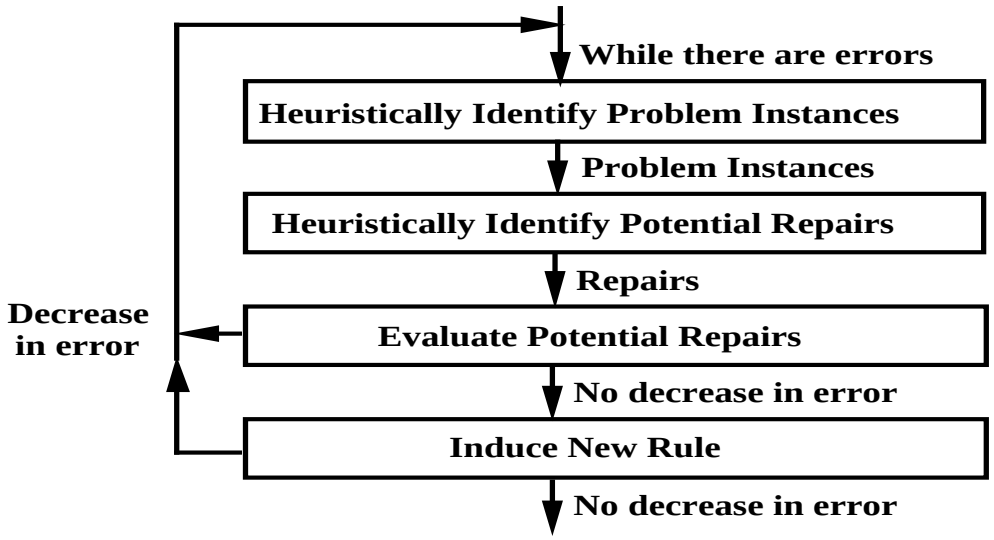

Figure 1: System Organization.

Rule-Base Evaluation

The metric used to evaluate of the rule-base (relative to a set of instances) is the mean error rate across all instances. The error rate for an instance is the percentage of constraints unsatisfied by the execution of the rule-base. An instance is executed in the following manner.

- Reset the rule-base.
- Assert any initial facts.
- Associate bindings with user-defined functions.
- Execute the rule-base until either the agenda is empty or until a user-defined rule execution limit is reached.

During execution of the rule-base for a particular instance, trace information is recorded that is used to determine how many of the constraints associated with the instance are unsatisfied.

Repair Operators

The set of potential repairs to a rule-base are, LHS specialization and generalization (the addition and deletion of conditional elements to the LHS of a rule), action promotion and demotion (the decrease and increase of the embeddedness of an action within if-than-else function, salience modification, assert and retract addition and deletion, observable action modification, rule deletion and rule induction.

Repair Identification

Below is a brief example of how repairs for a particular problem instance are identified. For a more thorough description of these and other aspects of CLIPS-R, see [5]. Assume a problem instance is identified that has an error because the final fact-list contains the extra fact *(repair "add gas")*. The heuristics that suggested repairs for an extra fact are described below.

- The action within the rule that asserted this fact should be deleted.
- Add a retract for the fact to a previously fired rule.
- For each unfired rule in the rule-base that has a retract that could retract this fact, identify repairs that would allow that rule to fire.
- Identify repairs that could cause the rule that asserted the fact to not fire.

EMPIRICAL ANALYSIS

A series of experiments were designed to analyze various characteristics of CLIPS-R. With the first rule-base, automobile diagnosis (distributed with CLIPS), we perform experiments to determine how well CLIPS-R does at increasing the accuracy of randomly mutated rule-bases. For the second domain, we deal with the nematode identification rule-base. With this rule-base, we show how CLIPS-R can be used to extend a rule-base to handle new cases. For the final rule-base, student loan [2], a problem translated from PROLOG to CLIPS, we chow that CLIPS-R is competitive with an existing revision system, FOCL-FRONTIER [6], that is designed to revise the Horn clause rule-bases.

Automobile Diagnosis Rule-Base

The auto diagnosis rule-base is a rule-base of 15 rules. It is an expert system that prints out an introductory message, asks a series of questions of the user, and prints out a concluding message including the predicted diagnosis.

Cases were generated from 258 combinations of responses to the user query function. Each instance consisted of a set of answers for each invocation of the query function as initial state information, a single constraint on the final fact-list and an ordering constraint for the sequence of printout actions. The target constraint for each instance was a positive constraint for a repair fact, e.g. *(repair "Replace the points")*.

Execution of an instance for the auto diagnosis rule-base consisted of clearing the fact-list, setting the bindings that determine the return values for each function call instance (to simulate user input for the user query function) and executing the rule-base to completion. The bindings that associate function calls to their return values allowed an otherwise interactive rule-base to be run in batch mode. This is necessary because

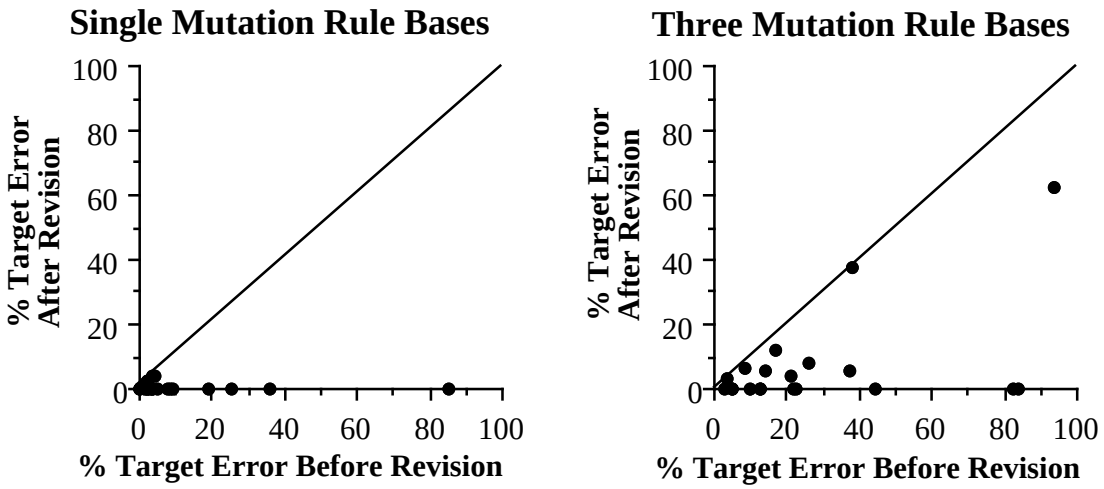

Figure 2: Target Error After Revision as a function of Target Error Before Revision.

no user would be willing to answer the same questions for 50 instances on different variations of the rule-base.

The experiments performed using the auto diagnosis rule-base were designed to determine how well CLIPS-R could do at revising mutated versions of the correct rule-base. Mutations consisted of extra, missing or incorrect conditional elements or actions and incorrect rule salience values. Two sets of 20 mutated rule-bases were randomly generated with one set of rule-bases having only a single mutation and the other set having three mutations per rule-base. Each mutated rule-base was revised using a random set of 50 training instances. The remaining instances were used for testing. Figure 2 contains a scatter plot showing the initial error of each mutated rule-base and the final error after revision of the rule-base.

An analysis of the scatter plots in Figure 2 shows that, for the most part, CLIPS-R is able to reduce the error rates of the mutated rule-bases (points below the diagonal indicate a decrease in error). For one mutation, the average rule-base error was 11.2% before learning and 0.7% after learning. With three mutations, the error before learning was 28.0% and after learning it was 7.2%.

Nematode Identification Rule-Base

The second rule-base, nematode identification (a nematode is a class or phylum of worm), has 93 rules. The intent in presenting this rule-base, is show an example of how CLIPS-R can be used to extend a classification rule-base to handle new cases. The basic requirements for this problem are a rule-base, a set of cases that the rule-base correctly classifies, and a set of cases that are not correctly classified by the rule-base.

For the nematode rule-base, because no cases were provided with the original rule-base, a set of 50 cases were generated by interactively running the rule-base over different

responses to the user query function. In order to simulate a rule-base that is in need of extension, two errors were introduced into the rule-base. Specifically, one rule was deleted (a rule that would normally assert the classification for two of the 50 cases), and a second rule was changed, so that it fired and asserted an incorrect classification for the two cases no longer classified by the deleted rule, see Table I. The two cases that are misclassified by the mutated rule-base, are the cases that CLIPS-R needs to extend the rule-base to cover.

Table I: Mutated Rules.

```
(defrule Pratylenchus
  ?f1 <- (esophagus-glands-overlap-intestine ventrally)
  ?f2 <- (ovary 1)
  =>
  (retract ?f1)
  (retract ?f2)
  (assert (nematode pratylenchus))
  (assert (id-criteria "1. esophagus glands overlap intestine ventrally."
                       "2. ovary 1."
                       "3. head-shape low and flat.")))
```

(a) Deleted rule.

```
(defrule Hirshmanniella
  ?f1 <- (esophagus-glands-overlap-intestine ventrally)
  ;;; ?f2 <- (ovary 2)
  =>
  (retract ?f1)
  ;;; (retract ?f2)
  (assert (nematode hirshmanniella))
  (assert (id-criteria "1. esophagus glands overlap intestine ventrally."
                       "2. ovary 2."
                       "3. head-shape low and flat.")))
```

(b) Rule with deleted conditional element *(ovary 2)* and retract.

When provided with the set of 50 cases and the mutated rule-base, CLIPS-R extends the rule-base to handle the new cases as follows. First, the two misclassified cases are identified by CLIPS-R as the problem cases. Second a set of repairs are identified and evaluated over all 50 cases. After completing the evaluations the repair that specialized the rule *Hirshmanniella* to include *(ovary 2)* as a conditional element is selected as the best repair because it is the repair that most decreased the error rate over all 50 cases. Upon permanently adding *(ovary 2)* to the rule *Hirshmanniella*, the two cases,

previously misclassified, are unclassified by the revised rule-base. After completion of a second set of repair evaluations with no success at reducing error rate, rule induction is successfully used to fix the unclassified cases, see Table II.

Table II: Revised Rules.

```
(defrule G123091
  (esophagus-glands-overlap-intestine ventrally)
  (ovary 1)
  =>
  (assert (nema-id pratylenchus)))
```

(a) New rule.

```
(defrule Hirshmanniella
  ?f1 <- (esophagus-glands-overlap-intestine ventrally)
  (ovary 2)
  =>
  (retract ?f1)
  (assert (nematode hirshmanniella))
  (assert (id-criteria "1. esophagus glands overlap intestine ventrally."
                       "2. ovary 2."
                       "3. head-shape low and flat.")))
```

(b) Revised rule with conditional element *(ovary 2)* added.

Note the difference between the original rules shown in Table I and the revisions of the mutated rules shown in Table II. The revised *Hirshmanniella* rule differs from the original rule by the absence of a retract for the fact matching the *(ovary 2)* conditional element. The set of 50 test cases were insufficient to recognize that a retract was missing. A similar problem is true for the induced rule *G123091*. This rule was was added by CLIPS-R to take the place of the the deleted rule *Pratylenchus*. While this rule asserts a classification that is correct with respect to the test cases, *(nema-id pratylenchus)*, it is not quite the same assertion made by the deleted rule, *(nematode pratylenchus)* (if *(nematode pratylenchus)* had been asserted by *G123091*, it would later be replaced by the fact *(nema-id pratylenchus)*). In short, the results of this experiment highlight the need for a comprehensive library of test cases.

Student Loan Rule-Base

In the original form, the student loan domain consists of a set of nine rules (represented as Horn clauses) and a set of 1000 cases. The rule-base contains four errors (an extra

literal, a missing literal, an extra clause and a missing clause). The initial theory has an error of 21.6%. In order to use this rule-base with CLIPS-R, the nine Horn clause rules were converted into nine production rules, each with a single assert action. Multiple clauses in the Horn clause rules were converted to a disjunction of conjuncts within a CLIPS production rule.

Execution of a case for the student loan rule-base consisted of asserting into an empty fact-list a set of facts specific to the case and then executing the rule-base to completion. All results for the following experiments are averages of 20 runs. All cases not used for training are used for testing.

Table III: A Comparison of FOCL-FRONTIER and CLIPS-R.

Num. Cases	FOCL % Error	CLIPS-R % Error
25	11.8	12.6
50	5.8	3.0
75	2.8	2.1

The experiment performed was to determine how well CLIPS-R performed at revising the rule-base relative to FOCL-FRONTIER. Table III shows that the error rate is competitive with that of FOCL-FRONTIER on this problem. Only with 50 training examples is the difference in error significant ($p < .05$).

FUTURE WORK

CLIPS-R is still in its infancy and we expect many of the details of the individual operators and heuristics to change as this work matures. Future directions include solutions to the issues that arose when revising the nematode rule, e.g. a better language for representing constraints on the correct execution of the rule-base, and the use of rule clustering, rule-models and rule-groups to guide the revision and induction of rules. Additional research could include a greater understanding of the distributions of rule-base coding styles, automated rule-base understanding systems, and revision strategies that simulate the methods by which humans manually revise rule-bases.

CONCLUSION

We have described CLIPS-R, a theory revision system for the revision of CLIPS rule-bases. Novel aspects of CLIPS-R include the ability to handle forward chaining theories with "nonlogical" operations such as rule saliences and the retraction of information from working memory. The system is organized with an iterative refinement control structure that identifies a set of similar problematic instances, identifies repairs that can fix the errors associated with the instances, and than evaluates each repair to identify

the repair that best improves the rule-base. CLIPS-R can take advantage of a variety of user specified constraints on the correct processing of instances such as ordering constraints on the displaying of information, and the contents of the final fact-list. In addition, CLIPS-R can operate as well as existing systems when the only constraint on processing an instance is the correct classification of the instance.

ACKNOWLEDGMENTS

The research reported here was supported in part by NSF Grant IRI-9310413, ARPA Grant F49620-92-J-0430, and AFOSR AASERT grant F49620-93-1-0569.

REFERENCES

1. Ourston, D. & Mooney, R., "Changing the Rules: A Comprehensive Approach to Theory Refinement," Proceedings of the Eighth National Conference on Artificial Intelligence, AAAI-90, 1990, 815-820.

2. Pazzani, M.J. & Brunk, C., "Detecting and Correcting Errors in Rule-Based Expert Systems: An Integration of Empirical and Explanation-Based Learning," In Knowledge Acquisition, 3, 1991, 157-173.

3. Wogulis, J. & Pazzani, M.J., "A Methodology for Evaluating Theory Revision Systems: Results with AUDREY II," Proceedings of the 13th International Joint Conference on Artificial Intelligence, IJCAI93, 1993, 1128-1134.

4. Clancey, W., "Classification problem solving," Proceedings of the National Conference on Artificial Intelligence, 1984, 49-55.

5. Murphy, P.M. & Pazzani, M.J., "Revision of Production System Rule-Bases," Machine Learning: Proceedings of the Eleventh International Conference, 1994, 199-207.

6. Pazzani, M.J. & Brunk, C., "Finding Accurate Frontiers: A Knowledge-Intensive Approach to Relational Learning," Proceedings of the Eleventh National Conference on Artificial Intelligence, AAAI93, 1993, 328-334.