In Proceedings of the Eighth International Machine Learning Workshop, pp. pp. 447-451, Evanston, IL, June 1991.

First-Order Theory Revision

Bradley L. Richards
Dept. of Computer Sciences
University of Texas at Austin
bradley@cs.utexas.edu

Raymond J. Mooney
Dept. of Computer Sciences
University of Texas, Austin
mooney@cs.utexas.edu

Abstract

Recent learning systems have combined explanation-based and inductive learning techniques to revise propositional domain theories (e.g., EITHER, RTLS, KBANN). Inductive systems working in first order logic have also been developed (e.g., CIGOL, FOIL, FOCL). This paper presents a theory revision system, Forte, that merges these two developments. Forte provides theory revision capabilities similar to those of the propositional systems, but works with domain theories stated in first-order logic.

1 INTRODUCTION

The past few years have seen a merger of inductive and explanation-based capabilities into a new class of systems performing *theory revision*. The premise of theory revision is that we can obtain a domain theory, be it from a book or an expert, but we cannot expect that theory to be entirely complete or correct. Theory revision systems use a set of training instances to improve the theory.

Forte (First-Order Revision of Theories with Examples) is a theory revision system for first-order logic. Theories are stated in a restricted form of Prolog, and a training set is used to identify where faults in the theory may lie. Forte uses operators drawn from propositional theory revision, first-order inductive systems, and inverse resolution to develop possible theory revisions.

2 RELATED WORK

2.1 PROPOSITIONAL THEORY REVISION

There are a number of propositional theory revision systems, including RTLS ([Ginsberg, 1990]), KBANN ([Towell, Shavlik, and Noordewier, 1990]), and EITHER ([Ourston and Mooney, 1990]). RTLS translates a theory into an operational form for use by an inductive learner, and translates it back into the theory language after modification. KBANN translates the initial theory into a neural network, and revises the network using standard neural network techniques. However, extracting a revised theory from the trained network is the subject of ongoing research. EITHER revises the theory directly, and in this sense is the most similar to Forte. However, all of these systems are limited to propositional domains. While most finite problem domains can be expressed in propositional form, doing so may greatly increase their size and reduce their understandability.

2.2 FIRST-ORDER LEARNING

Stephen Muggleton first pointed the way to first-order theory revision with his propositional system Duce ([Muggleton, 1987]). Duce uses six theory revision operators, which Muggleton later grouped under the heading *inverse resolution*. Duce takes advantage of the ease with which resolution steps can be reversed in propositional logic; if we know the resolvent and either the goal or the input clause, we can abduce the missing element. Duce uses an oracle to verify its operations.

From Duce it was a short but important step to CIGOL, a related system working in first-order logic ([Muggleton and Buntine, 1988]). CIGOL performs inverse resolution in first-order logic, but it assumes that all input clauses are unit clauses and, like DUCE, it depends on an oracle. Forte uses inverse resolution operators, but without these limitations.

[Quinlan, 1990] describes FOIL, an inductive learning system working in first order logic. FOIL works by generalization, constructing a set of Horn clauses that cover the positive examples while excluding the negative ones. FOCL, in [Pazzani, Brunk, and Silverstein, 1991], extends FOIL by using an input theory to guide and augment the search process. Clauses and portions of clauses from the input theory are considered for addition to the rules under development by FOIL. If adding a fragment of the input theory provides more information gain than adding a newly created antecedent, the term from the theory is chosen. Thus, providing a good input theory provides a substantial boost to the learning process. The primary difference between FOCL and Forte is that FOCL uses the input theory as an aid to the learning process, whereas Forte performs true theory revision.

3 PROBLEM DEFINITION

Our objective is to create a system that performs theory revision in first-order logic. The paragraphs below define our terminology, provide a more formal statement of our objective, and describe the restrictions placed on our prototype implementation.

3.1 THEORY

A theory, T, is a Prolog program without cuts.

3.2 ASSERTION

An assertion is a predicate corresponding with the consequent of one or more clauses in the theory. If an assertion is given that does not correspond to a clause in the theory, this indicates that a rule corresponding to the assertion needs to be added to the theory.

3.3 EXAMPLE

An example is a set of related instances that share a common set of facts. An example consists of a set of facts, F, a set of positive ground assertions, and a set of negative ground assertions. A fact is a ground atom corresponding to a predicate that may appear as an antecedent to clauses in the theory. A ground assertion is an instantiation of an assertion together with a boolean value indicating whether or not the ground assertion would be provable using a correct theory.

3.4 INSTANCE

An instance is a ground assertion with its associated truth value plus the facts associated with the example from which the ground assertion came. Instances that should be provable are *positive* instances, and instances that should not be provable are *negative* instances. Given a set, P, of positive instances and a set, N, of negative instances, we say theory T is *correct* on these instances if

$$\forall p \in P, T \cup F \vdash p$$
$$\forall n \in N, T \cup F \nvdash n$$

A training set $P \cup N$ is *consistent* if $P \cap N = \emptyset$. A theory cannot be correct on an inconsistent training set.

3.5 OBJECTIVE

Given an initial theory and a consistent set of instances, produce an "appropriately revised" theory that is correct on the given instances.

3.6 DISCUSSION

We say that a theory is appropriately revised if it meets certain heuristic criteria, namely

-- A revised theory should be as similar as possible to the initial theory, both semantically and syntactically.

-- A revised theory should be as simple as possible.

-- A revised theory should make meaningful generalizations from the input instances, so that it will be as accurate as possible on instances that did not appear in the training set.

3.7 RESTRICTIONS

The initial version of Forte does not allow recursion or negation in its theories, and it is vulnerable to local maxima, which means that it does not always generate a theory that is correct on the training set. Lifting the theory restrictions, limiting or eliminating Forte's susceptibility to local maxima, and providing a more formal definition of an "appropriately revised" theory are the primary goals of our ongoing research.

4 SYSTEM DESCRIPTION

Forte uses a training set to identify and correct errors and omissions in the given domain theory. It chooses, if they exist, one positive instance that is unprovable and one negative instance that is provable and proposes revisions to the theory that correct one of these errors. Each revision is evaluated globally, to see what its effect is on the theory's overall accuracy on the training set. The best revision is implemented, and the system chooses another pair of improperly classified instances.

Classes or categories in Forte are assertions that are to be proven using the domain theory. Training examples may include both attribute and relational information. Objects in training examples may be many-sorted, e.g., a domain might include both birds and bicycles, each with their own set of attributes.

The outermost layer of the program is a relatively simple iterative shell that calls the theory revision operators, evaluates the revisions they propose, and implements the best such revision.

Theory revision operators come from a variety of sources. Simple ones, like delete-rule, are drawn from propositional theory revision systems. Operators for adding and deleting antecedents are based on two separate derivations of FOIL. And operators for modifying intermediate rules are drawn from inverse resolution.

Operators can have three effects on a theory: specialization, generalization, and compaction. If a positive instance is unprovable then the theory needs to be generalized, whereas if a negative instance is provable the theory needs to be specialized. Forte also compacts (simplifies) the theory when doing so does not degrade accuracy on the training set. Operators implemented in Forte, and their effects, appear in Table I.

Note that, in several cases, Table I shows that an operator can be used both to compact the theory and to generalize or specialize it. In these cases, there are actually two versions of the operator. While conceptually similar, they work with different information toward different goals.

Table I. Operators may specialize, generalize, or compact a theory.

Operator	Spec	Gen	Com
Add Antecedent (FOIL)	✓		
Delete Rule	✓		✓
Delete Antecedent (Inverse FOIL)		✓	
Add Rule (FOIL)		✓	
Identification		✓	✓
Absorption		✓	✓
Delete Antecedent (ordinary)		✓	✓

Add antecedent (FOIL-based). If a negative example is provable, the proof may be forced to fail by specializing the theory. Each rule used in the proof is passed to a derivative of FOIL, along with sets of positive and negative instances. FOIL finds antecedents that distinguish between the positive and negative instances and adds these to the rule. If necessary, several rules will be added, each covering a portion of the positive instances.

Delete rule. If a negative example is provable, each of the rules used in the proof is considered for deletion from the theory. When used in compaction, a rule is deleted if doing so does not reduce the accuracy of the theory.

Delete antecedent (Inverse FOIL-based). If a positive instance is unprovable, each failing clause in the attempted proof is considered by delete-antecedent. This operator depends on a conceptual derivative of FOIL, called Inverse FOIL (IFOIL). Sets of positive and negative instances are passed to IFOIL, which deletes antecedents to create a rule allowing proof of some or all of the positive instances, but none of the negative ones. If necessary, IFOIL will create multiple rules to cover all positive instances.

Add rule (FOIL-based). If a positive instance is unprovable, each failure point in its proof is considered for add-rule. The failing clause is copied, with the failing antecedent deleted. If this allows the instance to be proven, FOIL is called to add any new antecedents that are required to keep negative instances from also becoming provable.

Identification (inverse resolution). Identification constructs an alternate definition for an antecedent identified in a failure point. It develops an alternate definition by performing an inverse resolution step using two existing rules in the domain theory. For example, suppose we need an alternate definition for predicate x, and we have the following two rules in the domain theory:

$$a \leftarrow b, x$$
$$a \leftarrow b, c, d$$

Identification will replace these two rules with the logically equivalent pair:

$$a \leftarrow b, x$$
$$x \leftarrow c, d$$

While this has no effect on the deductive closure of these rules alone, we have now introduced a new definition of x into the theory, which may allow our positive example to be proven. In first order logic, unification substantially complicates the picture, but the basic concept remains the same.

When used in compaction, identification seeks pairs of rules where it can construct definitions of intermediate predicates as shown. These changes are implemented if they reduce the size of the theory without reducing its accuracy.

Absorption (inverse resolution). Absorption is the complement of identification. Rather than constructing new definitions for intermediate predicates, absorption seeks to allow existing definitions to come into play. Suppose predicate c in the rule below is a failure point:

$$a \leftarrow b, c, d \qquad (1)$$

Now suppose our domain theory contains the following rule, as well as other rules with consequent x:

$$x \leftarrow c, d$$

In this case, absorption would replace rule (1) with the new rule

$$a \leftarrow b, x$$

thereby possibly allowing alternate definitions of x to be used when proving a.

In compaction, absorption makes the same kind of modifications to rules, trying to reduce the size of the theory without adversely affecting its accuracy.

Delete Antecedent (ordinary). The delete antecedent operator based on Inverse FOIL may be unable to develop a revision that excludes all negative instances. However, deleting an antecedent may still improve performance on the training set. Hence, this operator independently considers antecedents identified in failure points for deletion from the theory. When used in compaction, this operator will delete an antecedent if doing so does not degrade the performance of the theory on the training set.

5 RESULTS

In this section we present results showing Forte's learning performance on the family domain used in [Quinlan, 1990] to test FOIL. This gives us a basis for comparison to a first-order inductive learner. Readers familiar with FOIL should note that Forte's instance-based representation is substantially different from the tuple representation

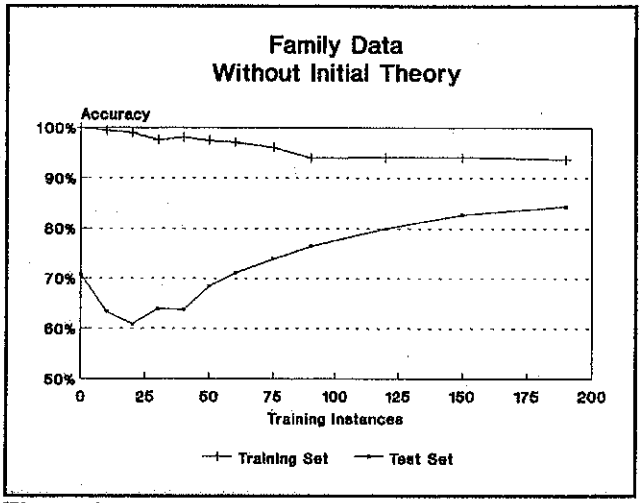

Figure 1. Forte performance on twelve family relations, with no initial theory. Data points are averaged over 21 trials.

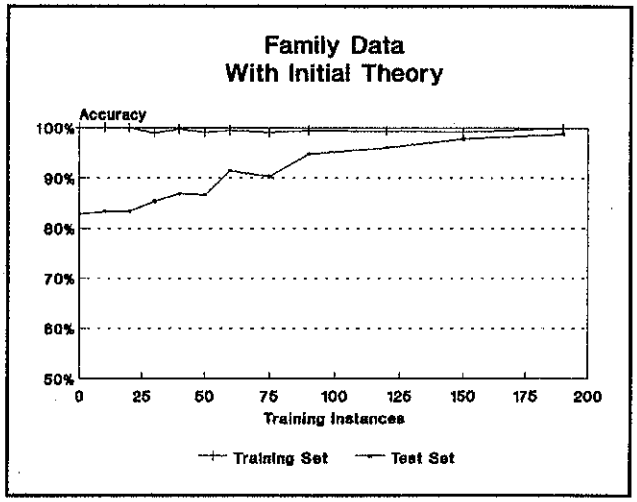

Figure 2. Forte performance on twelve family relations, using the initial theory shown in section 5.1 of the text. Averaged over 21 trials.

used by FOIL. FOIL used the equivalent of 2400 training instances to achieve 97.5% accuracy on this data. Forte's learning performance, both with and without an initial theory, is shown in Figures 1 and 2.

Training sets given to Forte were randomly selected from a database that included all 112 positive instances and 272 negative instances, which were chosen as being those closest to being provable. These are, in essence, the most useful negative instances to the theory revision process. FOIL, in its representation, had the equivalent of all positive instances and all negative instances that share their base constant with a positive instance (e.g., if John has an uncle, then FOIL would receive all negative instances of the form uncle(X, john)).

With no initial theory, Forte averaged 83% accuracy with 150 training instances, and improved slowly thereafter. With no training, we can reach 71% accuracy by guessing all instances to be negative. The initial fall-off in accuracy seen in Figure 1 reflects that fact that, with fewer than 75 instances, we do not have enough data for meaningful learning across twelve concepts. The training set performance shows that Forte is being caught in local maxima; in fact, with more than 75 instances, Forte rarely achieves 100% accuracy on the training set.

With an initial theory, Forte's performance improves dramatically. The given theory begins with an accuracy of 83%. Forte can completely correct the theory with as few as 120 training instances, and it rarely falls into local maxima. By the time Forte has seen 150 instances (an average of 12.5 per concept), training and test accuracies have nearly converged.

5.1 Revised theory

The theory below is the initial theory given to Forte in Figure 2. It contains multiple faults, including missing and added rules, missing and added antecedents, and incorrect antecedents. Added and incorrect items are shown in italics, and missing items are struck out.

wife(X, Y) :- gender(X, female), married(X, Y).
~~husband(X, Y) :- gender(X, male), married(X, Y).~~
mother(X, Y) :- gender(X, *male*), parent(X, Y).
father(X, Y) :- gender(X, male), parent(X, Y).
daughter(X, Y) :- gender(X, female), ~~parent(Y, X)~~.
son(X, Y) :- gender(X, male), parent(Y, X), *gender(Y, female)*.
sister(X, Y) :- gender(X, female), parent(Z, X), parent(Z, Y).
brother(X, Y) :- gender(X, male), parent(Z, X), parent(Z, Y).
brother(X, Y) :- parent(Z, Y).
aunt(X, Y) :- gender(X, female), aunt_uncle(X, Y).
uncle(X, Y) :- gender(X, male), *sibling(X, B), parent(B, Y)*.
niece(X, Y) :- gender(X, female), aunt_uncle(Y, X).
nephew(X, Y) :- gender(X, male), aunt_uncle(Y, X).
aunt_uncle(X, Y) :- sibling(X, B), parent(B, Y).
aunt_uncle(X, Y) :- married(X, A), sibling(A, C), parent(C, Y).
sibling(X, Y) :- parent(A, X), parent(A, Y), X \= Y.

Using 120 instances, Forte produced the correctly revised theory below, where additional compactions are shown in italics.

wife(X, Y) :- gender(X, female), married(X, Y).
husband(X, Y) :- gender(X, male), married(X, Y).
mother(X, Y) :- parent(X, Y), gender(X, female).
father(X, Y) :- gender(X, male), parent(X, Y).
daughter(X, Y) :- gender(X, female), parent(Y, X).
son(X, Y) :- gender(X, male), parent(Y, X).
sister(X, Y) :- gender(X, female), *sibling(X, Y)*.
brother(X, Y) :- gender(X, male), *sibling(X, Y)*.
aunt(X, Y) :- gender(X, female), aunt_uncle(X, Y).
uncle(X, Y) :- gender(X, male), aunt_uncle(X, Y).
niece(X, Y) :- gender(X, female), aunt_uncle(Y, X).
nephew(X, Y) :- gender(X, male), aunt_uncle(Y, X).
aunt_uncle(X, Y) :- sibling(X, B), parent(B, Y).
aunt_uncle(X, Y) :- married(X, A), sibling(A, C), parent(C, Y).
sibling(X, Y) :- parent(A, X), parent(A, Y), X \= Y.

6 CONCLUSION

Theory revision is an exciting development in machine learning, since it allows a system to take advantage of expert knowledge without requiring the expert to be infallible. In this paper we presented a system, Forte, that performs theory revision in first-order logic. Forte builds on prior work done in propositional theory revision, inductive learning, and inverse resolution. Future versions of Forte will lift a number of restrictions placed on the current system. Planned enhancements will introduce recursion and negation into the domain theories, and limit or eliminate Forte's susceptibility to local maxima.

Acknowledgements

This research was supported in part by the Air Force Institute of Technology faculty preparation program, and in part by the NASA Ames Research Center under grant NCC 2-629.

References

A. Ginsberg, "Theory Reduction, Theory Revision, and Retranslation," *Proceedings of the Eighth National Conference on Artificial Intelligence* (AAAI-1990), pp 777-782.

S. Muggleton, "Duce, an Oracle based approach to constructive induction," *Proceedings of the Tenth International Joint Conference on Artificial Intelligence* (IJCAI-87), pp 287-292.

S. Muggleton and W. Buntine, "Machine Invention of First-order Predicates by Inverting Resolution," *Proceedings of the Fifth International Conference on Machine Learning*, pp 339-352, 1988

D. Ourston and R. J. Mooney, "Changing the Rules: A Comprehensive Approach to Theory Refinement," *Proceedings of the Eighth National Conference on Artificial Intelligence* (AAAI-1990), pp 815-820.

M. J. Pazzani, C. A. Brunk, and G. Silverstein, "A knowledge-intensive approach to relational concept learning," *Proceedings of the Eighth International Workshop on Machine Learning*, 1991.

J. R. Quinlan, "Learning Logical Definitions from Relations," *Machine Learning*, 5:239-266, 1990.

G. G. Towell, J. W. Shavlik, and M. O. Noordewier, "Refinement of Approximate Domain Theories by Knowledge-Based Neural Networks," *Proceedings of the Eighth National Conference on Artificial Intelligence* (AAAI-1990), pp 861-866.