Relational clichés: Constraining constructive induction during relational learning

Glenn Silverstein and **Michael J. Pazzani**
silverst@ics.uci.edu and pazzani@ics.uci.edu
Department of Information and Computer Science
University of California, Irvine
Irvine, CA 92717 USA

Proceedings of the Eighth International Workshop on Machine Learning (pp. 203-207). Evanston, IL: Morgan Kaufmann.

Abstract

We discuss an approach to creating new terms during the induction of Horn clauses. The new terms enable a selective form of look-ahead during hill-climbing search. This look-ahead is needed because a conjunction of literals may be useful while each literal individually may not appear to be useful. We exploit knowledge of common patterns of conjunctions to avoid the necessity of testing all pairs of conjunctions.

1 INTRODUCTION

Relational concept learning has recently received a fair amount of attention, especially in the area of learning Horn clause descriptions of concepts (e.g., Quinlan, 1990; Muggleton & Feng, 1990). The basic problem is to learn a relational definition for a concept given a set of positive and negative examples of the concept and a set of known background predicates that can be used in the definition. The goal is to find some efficient means to search the space of possible concept definitions or to limit the space of concept definitions while still including useful concepts that may occur in a domain of interest.

FOIL (Quinlan, 1990) addresses this problem by using an information-based heuristic to guide a hill-climbing search. Although this has proven to be an effective means to learn a variety of concepts, there are a number of concepts which cannot be found using this search mechanism. For example, consider the trivial definition of `cup(X)` learned by many explanation-based learning systems (i.e., a cup is something that has a handle and an upward pointed concavity).

```
cup(X) :- partof(X,Y) & handle(Y) & partof(X,Z) & concavity(Z,up).
```

The predicate `partof(X,Y)` by itself is not likely to distinguish cups from non-cups (this would indicate that the number of parts of an object is important in determining whether the object is a cups) and hence would not be added to the definition for `cup(X)` by a hill-climbing algorithm. On the other hand, the conjunction `partof(X,Y) & handle(Y)` is more likely to distinguish cups from non-cups and hence, could be added[1]. The problem is thus reduced to finding and using combinations of predicates rather than just individual predicates. Unfortunately, trying all possible combinations of predicates is not a practical solution. Hence, a mechanism is needed to efficiently search through the space of combinations of predicates.

The solution we propose uses *relational clichés* to learn new predicates, such as `has_handle(X)`, which are defined in terms of existing predicates (e.g. `has_handle(X) :- partof(X,Y) & handle(Y))`. These clichés suggest potentially useful combinations of predicates through the use of patterns and restrictions that essentially open up a restricted subset of the space of combinations of predicates. By providing an efficient means of searching through a restricted subset of the space of combinations of predicates, relational clichés can increase the class of concepts that a relational concept learner can learn without significantly impacting the efficiency of the concept learner. In addition, the intermediate concepts produced in this process can be used to facilitate transfer. For instance, after `has_handle(X)` is formed while learning about cups, it may later be used in learning about briefcases. This paper explores the use of relational clichés in the context of FOCL (Pazzani & Kibler, 1990), which is an extension of FOIL. In principle, relational clichés should apply directly to FOIL and, perhaps with some adaptation, to other relational learners like GOLEM (Muggleton & Feng, 1990).

2 BACKGROUND

2.1 TERMINOLOGY

Throughout this paper the following terms will be used: *literal*, *clause*, *variabilization*, *extensional* and *intensional* definitions, and *tuples*. A *literal* is defined as a predicate with an associated set of variables, which are its arguments. *Clauses* consist of a single unnegated literal as the head of the clause and a conjunction of negated and unnegated literals as the body. A variable used in a literal in the body of a clause will be called "old" if it appears in the head or in an unnegated literal to the left of the current literal. Otherwise, a variable will be called "new". An *extensional* definition of a predicate

1. If FOIL had a different set of predicates that included predicates like `has_handle(X)` and `open_vessel(X)` to implicitly represent the conjunctions `partof(X,Y) & handle(Y)` and `partof(X,Z) & concavity(Z,up)` respectively, it could learn definition of cup more easily.

(concept) consists of a set of positive and negative examples. An *intensional* definition of a predicate is a set of one or more clauses that define the predicate in terms of other predicates. A *tuple* is a set of bindings for the old variables in a clause. For example, the clause `cup(X) :- partof(X,Y) & handle(Y)` consists of a head (`cup(X)`) and a body (`partof(X,Y) & handle(Y)`). `partof(X,Y)` is a literal in the body of this clause. `X` is an old variable in this literal (because it appears in the head) and `Y` is a new variable (since this is the first occurrence of `Y`). The clause itself represents a (possibly incomplete) intensional definition for `cup(X)`. The sets `{cup(c1),cup(c2),...}` and `{~cup(n1),~cup(n2),...}` provide an extensional definition of cup listing the positive and negative examples of cups respectively. The tuples associated with the extensional definition are: positive`{(c1),(c2), ...}`, negative`{(n1),(n2),...}`.

2.2 FOIL

FOIL (Quinlan, 1990) learns constant-free Horn clause theories that serve as intensional definitions of concepts. FOIL essentially starts out with a set of extensionally defined predicates, one of which is identified as the concept to be learned, and learns a set of clauses comprised of literals derived from the extensionally defined predicates. The learned clauses form an intensional definition of the concept. FOIL attempts to produce a set of clauses that satisfies all the positive tuples associated with the concept and none of the negative tuples. To accomplish this FOIL constructs new clauses until the set of clauses produced cover all of the positive tuples of the concept[2]. Clauses are constructed by adding literals to the current clause (which is initially empty) until the clause excludes all negative examples. A literal is added by trying possible variabilizations (i.e., ordering of old and new variables) of each predicate, and selecting the predicate and associated variabilization which maximizes an information theoretic heuristic called information gain.

2.3 FOCL

FOCL extends FOIL to take advantage of background knowledge. In particular, FOCL can compute the information gain of extensionally defined or intensionally defined predicates. Further, if an intensional definition is provided for the concept being learned (i.e., the target concept), FOCL can learn an operational definition for the target concept in terms of the extensionally defined predicates by operationalizing its intensional definition. This operationalization process is conducted by a modified explanation-based learning algorithm that operates on sets of positive and negative examples rather than a single example. FOCL uses FOIL's information gain metric to determine which path through a proof tree should be operationalized. Like FOIL, the end result of FOCL is an operational definition of the concept being learned which is in terms of the extensionally defined predicates provided.

Another addition that FOCL makes is the introduction of "primitive" predicates. For example, predicates such as < and >, need not be defined extensionally. Rather, FOCL defines these predicates as built-in primitives and can construct literals that compare an old variable to a real valued constant. The constant is selected to maximize information gain in a manner similar to that used by ID3 (Quinlan, 1986). This addition is used to define one of the relational cliché's described in the next section.

3 RELATIONAL CLICHES

Note that in the absence of a domain theory, FOCL acts like FOIL and attempts to select a predicate and an associated variabilization with maximal information gain. However, since FOIL uses a hill-climbing algorithm, if there are no predicates that have positive information gain, the algorithm simply fails. This is precisely what happens when FOIL tries to learn concepts like `cup(X) :- partof(X,Y) & handle(Y), ...` The main problem is that `partof(X,Y)` by itself just says that a cup has at least one part and is therefore not likely to have positive information gain. However, having a handle is likely to distinguish at least some cups from non-cup and hence, the conjunction `partof(X,Y) & handle(Y)` is likely to have positive information gain. In essence, `partof(X,Y) & handle(Y)` defines an instances of a class of semantically meaningful intermediate predicates that both FOIL and FOCL could use if they were defined extensionally or FOCL could use if they were defined intensionally (e.g., `has_handle(X) :- partof(X,Y) & handle(Y)`). Relational cliché's are an attempt to define these classes of predicates.

Relational cliché's consist of two parts: (1) a pattern, which is a abstract description of a conjunction of predicates (2) and a set of restrictions, which constrain the predicates that can be used to fill the associated pattern and the variabilizations that can be generated for the predicates of the pattern. Currently there are four predefined classes of predicates that can appear in a pattern: pred (all predicates of all types), ext-pred (extensionally defined predicates), comp (<,>, etc.), and arith-op (+,*, etc.). In addition, there are two types of-restrictions: predicate and variabilization restrictions.

Relational cliché's are used by FOCL in the following manner. Instead of failing when there are no literals with positive information gain, FOCL extends the search by computing the information gain of all combinations of predicates that are consistent with the relational cliché's. (See section 5 for a discussion of the complexity of this search.) If the combination of predicates with maximum gain has positive information gain, then it is added to the clause under construction. In addition, for some of the cliché's, a new intentionally defined predicate may be created to cache the results of instantiating the cliché. This new predicate can used in subsequent learning including the remainder of the current problem.

Listed below are four cliché's with an example of a conjunction of predicates derived from each one. The

2. The tuples associated with a clause are set initially to tuples of the extensional definition of the head of the clause (i.e., the concept being learned) and are extended to add any new variables introduced by the literals added to the body of the clause.

predicate and variabilization restrictions are omitted where they are obvious from the pattern. The philosophy behind the creation of these four cliché's and cliché's in general is similar to that behind CYC (Lenat & Guha, 1990): we are interested in efficiently handling commonly encountered problems. Hence, they have been constructed so that they are essentially domain independent and are meant to be widely applicable.

Partof cliché:

Pattern: `partof(A,B) & ext-pred(...,B,...)`

Restrictions: (1) the second predicate (`ext-pred`) must include the second variable (`B`) of the first predicate (`partof`). (2) `B` must be a new variable in the first predicate.

Although this cliché is currently restricted to include `partof` as the first element of the conjunction, we suspect this is just a specific case of a more general cliché in which the first predicate is one that permits certain inferences to transfer through it (Lenat & Guha, 1990). For example, ownerships transfers through `partof` (i.e., if you own `X`, you own the parts of `X`).

Example: `partof(X,Y) & handle(X).` Name:[3] `has_handle(X)`

Threshold Comparator cliché:

Pattern: `ext-pred(...,A,...) & comp(A,Thresh)`

Restrictions: (1) The first predicate (`ext-pred`) must introduce and bind a new variable that also appears as the first argument of the second predicate (`comp`) and is of a compatible type to the type required by the second predicate. (2) `Thresh` is a constant derived in a manner described in the previous section (i.e., derived from the value of `A` in the positive and negative tuples so as to maximize information gain)

Example: `blood-pressure(X,S,D),>(S,200).` Name: `high-blood-pressure(X)`

Arithmetic cliché:

Pattern: `D is A arith-op B & D comp C.`

Restrictions: (1) `D` must be a new variable of numeric type, (2) all other variables must be old and must be numeric, (3) `comp` must be an arithmetic comparator.

Example: `D is B+C & D>A.`
Name: `triangle-inequality(A,B,C)`

Recursive cliché:

Pattern: `reduction-pred({...,X,XNew})`[4] `& recursive-pred(...,XNew,...)`

Restrictions: (1)`reduction-pred` is a predicate which moves one of its argument closer to the base case of a recursively defined data type, e.g., `successor(XNew,X)` takes a numeric argument and moves it closer to 0 and `components(A,NewL,L)` takes a list `L` and returns it components `A` and `NewL` moving `NewL` closer to `[]`. (2) `X` must be an old variable (3) `recursive-pred` is the predicate being learned (4) `XNew` must occur in the same variable position as `X` in the head of the clause and (5) the remaining variables of `recursive-pred` must be equal to the corresponding variables in the head of the clause.

Example: `member(X,L) :- components(A,B,L) & member(X,B).`

The first two cliché's represent classes of "useful" intermediate concepts: the *partof* cliché essentially encodes all of the "has a part" concepts that can be associated with objects and other concepts like "has a small part" and the *threshold comparator* cliché includes those intermediate predicates that represent an extreme on the value of a predicate, e.g., "fast cars", "big trucks", "high-blood pressure", etc. When one of these two cliché's is instantiated and used, a new intermediate predicate can and should be created to cache the results so that it can be used later in completing the definition of the concept and in learning related concepts. Later on, a human can be consulted to provide the concept with a meaningful name. In this way, FOCL can gain transfer immediately and not have to relearn the instantiations of the appropriate cliché's.

The *arithmetic* and *recursive* cliché's are essentially "useful programming constructs". The arithmetic cliché is not powerful enough to allow FOCL to learn arbitrary arithmetic expressions, however, it can be used in conjunction with FOCL's domain theory to repair arithmetic expressions that are slightly inaccurate. The results of the *arithmetic* cliché can be cached as an anonymous concept to achieve the benefits of caching without forcing the user to name an unnatural concept (the instantiations of this cliché are not likely to be meaningful, except perhaps for the triangle inequality). The *recursive* cliché is meant to capture the regularity of naturally occurring recursive concept definitions. The basic idea is that recursive definitions tend to recurse over one or more arguments bringing the argument(s) (or a function of the argument(s)) closer to the base case. Caching the results of the recursive cliché does not make sense as they would only being useful for learning the same definition. However, in using its recursive cliché, FOCL need only consider combinations of predicates involving reduction predicates and the predicate being learned. The types of the arguments of the predicate being learned further restrict which reduction predicates can be used and once the reduction predicate(s) has been chosen all other variables are constrained. Hence, the recursive cliché is quite efficient and can allow FOCL to learn concepts it could not learn otherwise.

Relational cliché's give FOCL a selective look-ahead during learning that it can use when hill climbing fails. Without relational cliché's (and without a domain theory), FOCL cannot reliably learn concepts such as cups. In the implementation, FOCL only attempts to use cliché's if it reaches a point in which there is no literal with positive information gain. Once a cliché has been instantiated, a new intensionally defined predicate can be created. This

3. A mnemonic name need not be associated with a predicate.

4. Set notation is used to indicate that the cliché can be instantiated with the arguments in any order.

predicate can be used (and operationalized) by FOCL in the future, so that similar learning problems can now be solved directly with hill climbing (cf. Iba, 1989).

In the next section, we describe two experiments in which we compare FOCL with relational cliché's to FOCL augmented with a single unconstrained cliché that essentially tells it to try all conjunctions of two predicates. The two versions of FOCL are compared by counting the number of times they each compute the information gain of a literal[5]. The comparison should help illustrate how relational cliché's allow FOCL to learn concepts that FOCL alone is unable to learn with only a selective increase in the size of the hypothesis space searched. In Section 5, we provide a more formal theoretical analysis.

The unconstrained cliché is defined as:

Unconstrained cliché:

Pattern: `pred1(...) & pred2(...)`

Restrictions: None

4 EXPERIMENTAL RESULTS

4.1 THE CUP DOMAIN

The cup domain is adapted from Bergadano et al. (1988). The domain theory is listed below along with a typical positive and negative example. The domain theory is incomplete in that it is missing the intermediate concept `liftable`, whose correct definition is: `liftable(X) :- partof(X,Y) & handle(Y).`, i.e., a cup is liftable if it has a handle.

Domain Theory:

```
cup(X) :- stable(X) & open_vessel(X).

stable(X) :- partof(X,Y) & bottom(Y) & flat(Y).

open_vessel(X) :- partof(X,Y) & concavity(Y,up).
```

Positive Example:

```
a: partof(a,a1), bottom(a1), flat(a1), partof(a,a2),
body(a2), partof(a,a3), handle(a3),small(a3),
partof(a,a5), concavity(a5,up).
```

Negative Examples:

```
e: partof(e,e1), bottom(e1), flat(e1), partof(e,e2),
body(e2), small(e2), partof(e,e5), concavity(e5,up).
```

4.2 THE HIGH BLOOD PRESSURE/ELEVATED WHITE BLOOD CELL COUNT DOMAIN

The concept being learned in this artificial medical problem is `ill(X)`, where a person is ill if they have high blood pressure and an elevated white blood cell count. There is no domain theory for this problem. The predicates of the domain are `blood_pressure(P,S,D)`, `white_blood_cell_count(P,C)`, `street_address(P,S)`, `city_state(P,C,S)`. Two representative examples are:

Positive Example:

```
smith : blood_pressure(smith, 230, 90),
white_blood_cell_count(smith, 142),
street_address(smith,"22 foo st"), city_state(smith,
irvine, ca)
```

Negative Example:

```
jones : blood_pressure(jones,147, 88),
white_blood_cell_count(jones, 70),
street_address(jones,"2 bar st"), city_state(jones,
irvine, ca)
```

4.3 EXPERIMENTAL RESULTS

In the two domains, both versions of FOCL (i.e., FOCL + relational cliché's and FOCL + unconstrained cliché) were able to learn the correct definitions of the concepts. Table 1 shows the amount of work done by each algorithm on the two domains described above. In addition, the amount of work required to relearn the concept after constructive induction (CI) has added the new predicates it learns is shown.

For the cup domain, both FOCL + constrained relational cliché's and FOCL + unconstrained cliché relied on FOCL's explanation-based component to operationalize `stable(X) & open_vessel(X)` to produce the first part of the clause: `cup(X) :- partof(X,A) & bottom(A) & flat(A) & partof(X,B) & concavity(B,up)` and used constructive induction with their respective cliché's to complete the definition (constructing an appropriate intermediate term, i.e., `has-handle(X)`):

```
cup(X) :- partof(X,A) & bottom(A) & flat(A) &
partof(X,B) & concavity(B,up) & partof(X,C) &
handle(C).
```

The second problem had no domain theory so the two versions of FOCL needed to rely on constructive induction and two applications of their respective cliché's to arrive at a correct definition:

```
ill(X) :- blood_pressure(X,S,D) & S>190 &
white_blood_cell_count(X,C) & C > 10,600.
```

Table 1. Number of literals tested

problem	constrained	unconstrained	after CI
cup(X)	171	5158	115
ill(X)	139	1577	63

5 DISCUSSION

In the experiments above, the additional work required to employ the constrained cliché's was small compared to the amount of addition work required to use the unconstrained cliché. Using the constrained cliché's, FOCL required 1.5 and 2.2 times more work than with the appropriate intermediate concepts predefined for the `cup(X)` and `ill(X)`) domains respectively (i.e., `has-handle`, etc.), whereas, FOCL required 45 and 25 times more work to learn the same concepts using the unconstrained cliché's in the same domains. The exact amount of additional work required to implement a cliché depends upon the particular domain (i.e., the arity and number of known predicates) and the particular clause being learned (i.e., the number of old variables). However, the analysis of FOIL provided by Pazzani and Kibler (1990) can be

5. Pazzani and Kibler (1990) argue that this is a good metric to measure the search space explored by FOCL.

extended to give a bound on the search required by each cliché. Pazzani and Kibler show that the cost of considering all variabilizations of all predicates is bounded by $p(o+a\text{-}1)^a$ where a is the largest arity on any predicate and there are o old variables in the clause and p total predicates. For the clichés, $p^2(o+a\text{-}1)^a(o+2a\text{-}2)^a$ is an upper bound on the number of ways of instantiating the unconstrained cliché (since this is a conjunction of two predicates each of which must have at least one old variable). In contrast, with e extensionally defined predicates, m arithmetic operators, and c arithmetic comparison predicates, there are at most $oea(o+a)^{a\text{-}1}$ ways of instantiating the *Partof* cliché, o ways of instantiating the *Recursive* cliché, $c(e(o+a\text{-}1)^a - e(o)^a)$ ways of instantiating the *Threshold Comparator* cliché and cmo^3 ways of instantiating the the *Arithmetic* cliché.

As a result of this analysis, we would expect the benefits of relational clichés on larger problems to be even more substantial. In additional, the cost of using the predicates added as a result of instantiating a cliché in future learning is minimal. In particular, the cost of examining all predicates plus the newly added predicate defining an instantiated cliché (e.g., `has_handle(X) :- partof(X,Y) & handle(Y)`) is at most $(p+1)(o+a\text{-}1)^a$.

6 FUTURE DIRECTIONS

The above discussion indicates that relational clichés can increase the class of concepts that can be learned without a major effect on the computational complexity of the resultant algorithm. In particular, the constraints imposed are an alternative to those proposed by Muggleton and Feng (1990), and permit some concepts to be learned with literals that are not determinate (e.g., `partof(X,Y)` is not determinant).

Although the size of the search space explored is only linear in the number of predicates, it is possible that creating large numbers of new intentionally defined predicates can slow down the learning process. To overcome this problem, the newly created predicates could be monitored to determine when the cost of checking them exceeds the cost of rederiving them using their respective clichés. When the cost of checking a seldom used predicate exceeds the cost of rederiving it, that predicate can be deleted.

In addition, our future directions include:

- Testing out clichés in large domains
- Exploring methods to automate the process of developing clichés through learning.

To address the first issue we plan to run FOCL with the four clichés described in this paper on other more realistic domains. We anticipate adding a small number of additional clichés by hand, and perhaps generalizing some of the existing clichés. An example of a cliché that might be added is one that conjoins a determinate term with another literal.

To address the second issue, we plan to investigate a number of approaches to automatically creating clichés. One approach would be to examine existing Horn clause theories, either those created automatically by relational learning programs, or those coded by hand (e.g., expert systems), and find patterns among the rules. Another approach we intend to investigate is the use of learning techniques that attempt to specialize the unconstrained cliché. The idea here is to start with one very general cliché and identify commonly used specializations of this cliché. This approach is similar to the approach used in Pazzani (1991) to find common patterns of causal relationships.

Acknowledgements

We would like to thank Ross Quinlan for advice on FOIL, Dennis Kibler and David Aha for advice on the complexity of relational learning, and Elenita Silverstein, Caroline Ehrlich, and Virginia Silverstein for commenting on an earlier draft of this paper. This research is supported in part by NSF grant IRI-8908260.

References

Bergadano, F., & Giordana, A. (1988). A knowledge intensive approach to concept induction. *Proceedings of the Fifth International Conference on Machine Learning* (pp. 305–317). Ann Arbor, MI: Morgan Kaufmann.

Iba, G. A. (1989). A heuristic approach to the discovery of macro-operators. *Machine Learning, 3,* 285–317.

Lenat, D., & Guha, R.V. (1990). *Building large knowledge-based systems: Representation and inference in the CYC project.* Reading, MA: Addison-Wesley.

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.

Pazzani, M., & Kibler, D. (1990). *The utility of knowledge in inductive learning* (Technical Report No. 90-18). Irvine: University of California, Department of Information & Computer Science.

Pazzani, M. (1991). *Theory-driven learning: Using intra-example relationships to constrain learning.* (Technical Report No. 91-07). Irvine: University of California, Department of Information & Computer Science.

Quinlan, J.R. (1986). Induction of decision trees. *Machine Learning, 1,* 82–106.

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