Traps and Pitfalls when Learning Logical Theories: A Case Study with FOIL and FOCL

Giovanni Semeraro
semeraro@vm.csata.it
Dipartimento di Informatica
Università degli Studi di Bari, Via Orabona 4, 70126 Bari, Italy

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

Technical Report 93-33
July 26, 1993

This work has been carried out when Giovanni Semeraro was visiting the machine learning research group of Prof. Michael Pazzani at the University of California, Irvine.
The authors would like to thank Lorenza Saitta, Kamal Ali, and Luc De Raedt for constructive criticism on this work, Cesare Tinelli for the help in proving Proposition 1, Piew Datta for commenting on an earlier draft of this paper, and Peter Idestam-Almquist for suggesting Example 2 and for helpful discussions on the formal topics of this work. Thanks also to all the graduate students of the Artificial Intelligence Research Center at the University of California, Irvine for helpful comments on the ideas presented in this paper. Document understanding data set was provided by Enrico Annese from Olivetti Systems & Networks.
A special thank to Prof. Floriana Esposito for supporting Giovanni Semeraro during his stage in Irvine and for continuously stimulating his research and to Donato Malerba for several discussions on the topics of this paper.

To Ottavio, Sergio and Patrizio

Traps and Pitfalls when Learning Logical Theories: A Case Study with FOIL and FOCL

Giovanni Semeraro
Dipartimento di Informatica
Università degli Studi di Bari
Via Orabona 4, 70126 Bari, ITALY
(semeraro@vm.csata.it)
+39 (80) 243269

Clifford A. Brunk **Michael J. Pazzani**
Department of Information and Computer Science
University of California, Irvine
Irvine, CA 92717 USA
(brunk@ics.uci.edu)
(pazzani@ics.uci.edu)
(714)856-5888

Abstract

Supervised concept learning based on Horn clauses is one of the most active areas of machine learning research. Two popular systems in this area are FOIL (First Order Inductive Learner) and FOCL (First Order Combined Learner). This paper points out some conceptual traps and pitfalls in which these two systems fall when they cope with some tasks taken both from the machine learning literature and from real world domains. An interpretation of the obtained results is provided. It is based on a comparison between the search space that these two systems should explore and the search space that they actually explore and on considerations about the representation of the examples as tuples in a relational database. Theoretically-founded solutions to the detected problems are suggested. Moreover, a more manageable practical solution is proposed and its strengths and weaknesses in comparison with the theoretically-founded ones are evaluated. Such a solution has been satisfactorily implemented in a new version of FOCL.

Contents

1. Introduction

Inductive learning is undoubtedly the paradigm that has been most widely investigated in machine learning. It can be subdivided into *learning from examples* - also called *concept acquisition* or *supervised learning* - and *learning by observation and discovery* or *unsupervised learning* (Carbonell *et al.*, 1983). In the former strategy, given a set of examples and counterexamples of one or more concepts, the task is to determine a general description - also called *concept definition*, *predicate definition*, *rule*, or *inductive hypothesis* - for each concept to be learned that explains all of the positive examples and none of the counterexamples. In the latter strategy, the task is to find regularities explaining a collection (or at least most) of unclassified observations.

Recently, a new research area, called Inductive Logic Programming (ILP) (Muggleton, 1991), was born. It lies between logic programming and learning from examples, and aims at inductively learning *logic programs* (Apt, 1990) from examples. ILP is characterized by a formal approach to the problem of learning from examples.

The first formal method developed for *inductive generalization* is due to Plotkin (Plotkin, 1970; Plotkin 1971a; Plotkin, 1971b). Most of the work in the area of ILP refers to Plotkin's framework in order to define the space in which the search for concept descriptions is performed. Unfortunately, the properties of the lattice $\mathfrak{C}$ defined by Plotkin (1970) considering full clausal logic are not preserved when the proper subset of Horn clause logic is considered. In particular, Plotkin's operator for computing the *greatest lower bound* (*glb*) of two clauses is not closed for Horn clauses, thus the set of Horn clauses is not a sublattice of $\mathfrak{C}$. As a consequence, the use of Plotkin's framework for definite Horn clauses as in much of the machine learning literature is at least imprecise.

Shapiro (1981) presented a framework for the inductive inference of logic theories. He developed an incremental inductive inference algorithm, the *Model Inference Systems* (MIS), that can identify in the limit (Gold, 1967) any *model* in a family of complexity classes of first order theories. Both Plotkin's and Shapiro's systems proved very inefficient on real world tasks and were able to infer only simple logic programs from a small number of examples.

Helft (1987) proposed a logical framework for inductive generalization that aims at reducing the size of the search space by restricting the representation language to function-free (but not constant-free) clausal logic. The two constraints adopted by Helft, namely the *difference links* and the *linkedness* of the Horn clauses, allow his algorithm (for finding most specific generalizations) to infer better generalizations from a practical point of view. Moreover, Helft's algorithm does not initially assume the difference links between variables, but *learns* all and only those difference links necessary to determine the *least general generalization* (*lgg*) of two examples. More explicitly, it finds the lgg of two clauses with respect to an equality theory with an axiom schema of the form $t_i \neq t_j$ for $i \neq j$, where t_k, k=1,2,..., denotes either a constant or a variable of the language.

The second constraint reduces both the size and the scope of the search space, but Helft claims that it prevents his algorithm from finding meaningless rules.

Helft's framework, together with Plotkin's, is now commonly recognized as one of the two main frameworks for ILP (Bell and Weber, 1993; Kietz, 1993).

Both the language biases adopted by Helft were explicitly embodied with some differences in the INDUCE program (Larson, 1977; Michalski, 1980), although the terminology used is completely different. The equivalent of the difference links is obtained by using a particular existential quantifier called *distinct existential quantifier* and replacing the concept of θ-subsumption with the slightly different concept of subgraph isomorphism, while linked clauses are called *completely connected* graphs (Larson, 1977). It is worthwhile to note that INDUCE cannot learn Horn clauses but only decision rules in which variables are not allowed to occur in the action part.

Recently Quinlan (1990) has developed a system, called FOIL (First Order Inductive Learner), that learns definite Horn clauses from data expressed as relations. FOIL proved effective and efficient on many real world tasks. Nevertheless, Quinlan himself recognizes that *"it is not difficult to construct tasks on which the current version of FOIL will fail"*. His interpretation is that the main reason for this failure is the greedy search performed to add literals to the body of each clause in a concept definition.

FOCL (First Order Combined Learner) (Pazzani and Kibler, 1992) is an extension of FOIL in several aspects. The main extension allows FOCL to define and exploit background knowledge in the inductive learning process.

Both FOIL and FOCL are widely acknowledged as an advance in the area of learning from examples (Wirth and O'Rorke 1991; Markov, 1993) and some authors recently provided FOIL with a theoretical foundation by casting it in the Plotkin's framework (Bell and Weber, 1993).

This paper aims at pointing out some conceptual traps and pitfalls in which both FOIL and FOCL fall when they cope with both toy world problems taken from the machine learning literature and real world tasks taken from different domains. Furthermore, it provides a novel interpretation of the behaviour of these systems based on a pure analytical approach to the problem of concept learning. With this purpose, Section 2 presents a brief overview of FOIL and FOCL and describes the representation language used by these two systems. A critical review of Plotkin's framework for the purposes of concept learning is presented in Section 3. Furthermore, we define a new operator $\wedge_{\mathcal{T}}$ and prove in Section 3.2. that it computes the glb of any two Horn clauses, thus making the set of Horn clauses a lattice. Negative experimental results concerning FOIL and FOCL are shown in Section 4. All these negative results are explained by using the presented logical framework. These explanations straightforwardly suggest the way for overcoming the detected conceptual problems. The general theoretically-founded solutions are proposed in Section 5. In the same section, we claim that a language bias, namely *object identity*, achieves the same effect as the theoretical solutions in most but not all the cases. Moreover, we point out that object identity corresponds to introduce a model of generalization weaker than θ-subsumption, which increases the *size* of the search space defined by a generalization hierarchy by causing the splitting of each equivalence class into an infinite number of distinct equivalence classes, but reduces the *scope* of the search. It is worthwhile to note that object identity is equivalent to the distinct existential quantifier, denoted by $\exists.$ in INDUCE (Larson, 1977) and INDUBI (Semeraro, 1988; Esposito, 1990) and by $\exists^*$ in (Haussler, 1989), one-to-one substitutions in SIERRA (VanLehn, 1989), and *object-oriented generalization* in MAGGY (Manago and Kodratoff, 1987). As already noted in Dietterich and Michalski (1983), by dropping this constraint, *"the algorithm* - N.d.R. THOTHP (Vere, 1980) - *creates generalizations involving a many-to-one binding of variables. While such generalizations may be desirable in some situations, they are usually meaningless, and their uncontrolled generation is computationally expensive"*.

2. FOIL and FOCL

This section presents a brief overview of FOIL and FOCL.

2.1. FOIL

FOIL (Quinlan, 1990) combines the separate-and-conquer strategy adopted by the AQ family of learning systems (Michalski, 1969; Larson and Michalski, 1975; Michalski and Larson, 1983) with the information-based heuristic used by some systems that learn decision trees (Quinlan, 1986).

In FOIL each k-ary predicate is associated with a *relation* consisting of the set of k-tuples of constants

that satisfy that predicate. Thus, for instance, the binary predicate *on-top* will be associated with the following relation:

on-top
<a_3, a_5>
<a_3, a_7>
<a_5, a_6>

where a_3, a_5, a_6, a_7 are constants used to represent distinct objects. Of course, all occurrences of a given constant in all the relations defined for a given learning problem denote the same subpart. A relation is associated to the predicate that we want to learn, as well. Such a relation is called *target relation*. Each relation, including the target relation, defines both *positive* and *negative instances* of the predicate, so that if a_3 and a_6 are not in the relation *on-top* the tuple $<a_3, a_6>$ can be defined as a negative instance.

Negative instances of a predicate can be defined either explicitly or according to the *closed-world assumption* (CWA) of Prolog. In this last case negative instances consist of all constant k-tuples other than those considered positive and they are generated by temporarily making a *domain closure assumption* (DCA).

Given a Prolog clause of the form $P(X_1, X_2, ..., X_k) \leftarrow \varphi(V_1, V_2, ..., V_r)$, $V_i \in \{X_1, X_2, ..., X_k\} \cup \{Y_1, Y_2, ..., Y_n\}$, i=1,2,...$r$, where $\varphi(V_1, V_2, ..., V_r)$ denotes a conjunction of literals and the Y_i's being new variables introduced by at least one of the literals in $\varphi(V_1, V_2, ..., V_r)$, then the set of constant $(k+n)$-tuples such that $\varphi(V_1, V_2, ..., V_r)$ is satisfied can be partitioned into two sets as well: *positive* and *negative tuples*. Positive tuples are those for which their projections on the set of variables $X_1, X_2, ..., X_k$ define positive instances of P while the others are negative tuples. Therefore, if we want to learn the predicate *top-part* and the corresponding relation is:

top-part
<a_5>
<a_7>
<c_6>

then the clause *top-part(X)* $\leftarrow$ *on-top(X,Y)* is satisfied by the set of tuples associated to the *on-top* relation:

X Y
<a_3, a_5>
<a_3, a_7>
<a_5, a_6>

where the last one is a positive tuple while the first two are negative since a_3 is not a positive instance for the predicate *top-part*.

Given a k-ary predicate P, FOIL learns a predicate definition or rule for P, that is a set of definite Horn clauses in which $P(X_1, X_2, ..., X_k)$ is the literal in the head. Each clause of this rule should be satisfied only by positive tuples while all clauses together should cover all positive instances of $P(X_1, X_2, ..., X_k)$. In other words, FOIL seeks a rule that is complete and consistent.

FOIL adopts a separate-and-conquer search strategy to build a set of clauses. It starts with the unit clause $P(X_1, X_2, ..., X_k) \leftarrow \varnothing$ whose body is empty and adds literals one by one to the body until it gets a clause of the form $P(X_1, X_2, ..., X_k) \leftarrow L_1, L_2, ..., L_n$ that is consistent. Then, positive instances of P that are covered by such a clause are removed from the training set and, if there are still positive instances to cover, a new clause is learned starting again from the unit clause.

The decision of the literals L_i to be added is made according to an information theoretic heuristic called *information gain* and defined as follows:

$$Gain(L_i) = T^{++} \cdot [\log_2(p_1/(p_1+n_1)) - \log_2(p_0/(p_0+n_0))]$$

where:

- p_0 and n_0 are the numbers of positive and negative tuples covered by the clause *before* adding the literal L_i, respectively
- p_1 and n_1 are the numbers of positive and negative tuples covered by the clause *after* adding the literal L_i, respectively
- T^{++} is the number of positive tuples covered by the clause before adding the literal L_i that have at least one corresponding extension in the positive tuples after adding the literal.

Note that $Gain(L_i)$ is negative if the positive tuples are less concentrated in the new set of tuples covered after adding L_i, than in the current set of tuples. Among all the literals having positive gain, FOIL chooses that maximizing the gain function. But for which literals does FOIL test the corresponding gain? Each literal L_i in the body of a clause takes one of the four forms $X_j = X_k$, $X_j \neq X_k$, $Q(V_1, V_2, ..., V_t)$, $\neg Q(V_1, V_2, ..., V_t)$, where the X_i's are variables already introduced by previous literals (*old* variables) in the body, the V_i's can be both old and *new* variables, and Q is the same predicate associated with a relation. Therefore, FOIL learns *function-free* Horn clauses since terms other than variables cannot occur as arguments of a literal.

Some restrictions are imposed:

1) the literal must contain at least an old variable
2) infinite recursion has to be avoided when $Q \equiv P$.

The first restriction is due to the fact that we are interested to learn only *linked Horn clauses* (Helft, 1987). Furthermore, unlinked literals do not have gain anyway.

Definition 1 (Linkedness) A Horn clause is *linked* if all of its positive literals are linked. A positive literal is linked if at least one of its arguments is linked. An argument of a literal is linked if either the literal is the head of the clause or another argument in the same literal is linked.

In FOIL, a Horn clause is linked if both its positive and its negative literals are. Linkedness is a restriction on the arguments of a clause that reduces both the size and the scope of the search space. Nevertheless, it generally prevents a learning system from introducing meaningless literals in the body of the learned clause. Therefore, similar restrictions (sometimes exactly the same restriction) are adopted by many researchers and can be found in (Larson, 1977) with the name of *connected formulas*, in (Vere, 1980) with the name of *association chains*, in (Buntine, 1988) and (Manago, 1986).

Of course, one might object that the clause $p(X) \leftarrow r(Y), q(X,Y)$ is linked but the first literal, $r(Y)$, contains only new variables. However, it should be observed that such a clause is equivalent to $p(X) \leftarrow q(X,Y), r(Y)$, which is, generally, computationally more efficient than the previous one. Moreover, it should be observed that the body of the clause $p(X) \leftarrow \neg r(Y), q(X,Y)$ must be interpreted as $\neg (\exists Y\ r(Y)) \wedge \exists Y\ q(X,Y)$, while $p(X) \leftarrow q(X,Y), \neg r(Y)$ must be interpreted as the very different $\exists Y\ (q(X,Y) \wedge \neg r(Y))$ according to the standard Prolog interpretation of negated literals based on the negation-as-failure rule (Clark, 1978). In other words, when a negated literal $\neg r(Y)$ introduces only new variables, the whole clause will never be satisfied if there exists at least one element in the relation associated with r.

As to restriction 2), it should be pointed out that Horn clauses may cause infinite recursion when some predicate is allowed to appear both in the head and in the body of a clause. In order to prevent this, Quinlan introduces a technique based on the concept of *irreflexive partial ordering* between variables. It is worthwhile to point out that this technique does not guarantee finite recursion on previously unseen examples, that is, on new tuples in the relation associated to a predicate defined by a recursive clause. In addition, such a recursion check is not useful for mutual recursion (neither for the tuples in the training set).

Finally, we observe that the search strategy used by FOIL while building a single clause is a hill-climbing strategy which suffers from the *horizon effect*: A literal that proves to be desirable or even

essential from a global perspective (the whole clause) may appear relatively unpromising at a local level (a partially developed clause) and so may be left out. In order to avoid this problem, Quinlan (1991) introduced the concept of *determinate* literal. Suppose that we have an incomplete clause $A \leftarrow L_1, L_2, ..., L_{m-1}$ with an associated set T of tuples. A literal L_m is said to be *determinate* with respect to this partial clause if L_m contains new variables and there is exactly one extension of each positive tuple in T, and no more than one extension of each negative tuple that satisfies L_m. Unless a literal is found whose gain is close to ($\geq 80\%$ of) the maximum possible gain, FOIL adds all determinate literals to the clause and tries again. The effect is that the search space is enlarged. At the end of the learning process, when a predicate definition has been generated, a post-processing of clauses will prune all those literals, determinate or not, that will turn out to be unnecessary or even counterproductive.

2.2. FOCL

FOCL (Pazzani and Kibler, 1992) is an extension of FOIL in several aspects. In particular, some extensions allow FOCL to prune the search space while other extensions allow FOCL to define and use background knowledge for the problem at hand.

A difference between these two systems concerns the use of typing information that allows FOCL to reduce the search space. For instance, in the well known artificial task of learning to classify trains as Eastbound or Westbound (Larson, 1977), suppose that the partial clause *eastbound(X)* ← *has-car(X,Y)* has been generated.

Then, FOCL will not test any *variabilization* of a predicate, say *infront*, in which X occurs - that is, literals *infront(X,Y)* or *infront(Y,X)* - when it searches for a new literal to add. The type of the first argument of the predicate *has-car* is *train* while the type of the second argument is *car*. This means that X and Y have different types but both arguments of the predicate *infront* are of type *car*. Therefore, *infront(X,Y)* and *infront(Y,X)* are meaningless with respect to typing, so that they should not be tested. FOIL does not exploit such a typing information, so wasting time to test meaningless literals[1].

As to the use of background knowledge, we have already said that FOCL allows the definition of both *extensional* and *intensional* predicates. A predicate is defined extensionally when a relation (set of tuples) is associated with it, while a predicate is defined intensionally when it appears in the head of an inference rule expressed as a Horn clause. Alternative terms to denote extensionally and intensionally defined predicates are *operational* and *non-operational* predicates respectively. The goal of FOCL, like FOIL, is to create a predicate definition or rule in terms of the operational predicates[2], that is complete and consistent. During the search for such a rule, FOCL manages operational and non-operational predicates in the same way since the information gain can be computed for predicates defined both extensionally and intensionally. Moreover, when a partial, possibly incorrect, intensional definition of the concept to learn is provided, FOCL uses the information gain metric in order to operationalize the concept description as in explanation-based learning.

Another characteristic that distinguishes FOCL from FOIL is the availability of *relational clichés* that suggest potentially useful combinations of literals to test while generating a clause of a predicate definition (Silverstein and Pazzani, 1991). In this way, clichés provide a form of look-ahead that tries to overcome the problem of horizon effect leading a hill-climbing search strategy to find local rather than global maxima. Relational clichés are an alternative approach to *determinate literals* exploited by FOIL.

One cliché is the *non-numerical-constant* cliché. It enables FOCL to learn Horn clauses in which

1. Updated versions of FOIL, namely FOIL2 through FOIL5.0, include type definitions in order to overcome this problem.
2. A more recent version of FOCL also allows to find rules in which non-operational predicates appear in the body of the single clauses. In this case, the background knowledge is necessary in the classification process too.

constants can appear in the body of a clause by testing the pairs of literals $Q(X, Y)$, $eql\text{-}value(Y,y_i)$, where Q is a predicate while y_i is one of the possible values that Q can take. Note that y_i is a constant, thus FOCL is not constrained to learn function-free Horn clauses as FOIL when the non-numerical-constant cliché is used. The definition of such a cliché may have some undesirable side effect, so that it may be sometime preferable to constrain the language of hypotheses to that of function-free Horn clauses.

Finally, it should be observed that FOCL implements a very simple technique to prevent infinite recursion. In particular, when a recursive literal $p(Y)$ is added to the body of a clause whose head is $p(X)$, the literal $not(X=Y)$ is added as well. This technique is better than that implemented in mFOIL (Dzeroski and Bratko, 1992), which does not allow a clause to contain a literal in its body that is identical to the literal in its head, but less sophisticated than that implemented in FOIL.

3. A critical review of Plotkin's framework

3.1. Preliminaries

Subsequently, we refer to (Lloyd, 1987; Apt, 1990) for what concerns the basic definitions of a *substitution*, *most general unifier* (*mgu*), *positive* and *negative literal*, *clause*, *definite* (or *program*) *clause* and *Horn clause*. In particular, we denote a substitution (or *variable binding*) σ with

$$\sigma = \{ x_1 \leftarrow t_1, x_2 \leftarrow t_2, \ldots, x_n \leftarrow t_n \}$$

where the terms $t_1, t_2, \ldots, t_n$ replace the variables $x_1, x_2, \ldots, x_n$, respectively. Furthermore, $dom(\sigma)$ denotes the following set:

$$dom(\sigma) = \{ x_1, x_2, \ldots, x_n \}.$$

Given a first order expression ϕ, $vars(\phi)$ is the set of the variables occurring in ϕ. We will consider a clause as the set of its literals. If C_i is a definite Horn clause, then we will denote with H_i its head. Any two clauses are always assumed to be variable disjoint. This does not limit the expressiveness of the adopted language since any two *non variable disjoint* clauses always can be standardized apart.

Given two expressions ϕ_1 and ϕ_2, $MGU(\phi_1, \phi_2)$ is the set of all the (equivalent) mgu's of the expressions. We denote with $\equiv$ the syntactic equality between two expressions. Henceforth, we denote with $\mathfrak{C}$ the set of the clauses of the language and with $\mathfrak{H}$ the subset of Horn clauses. The term *logic theory* (or simply *theory*) is used as a synonym of *logic program*, i.e. a set of definite Horn clauses.

Definition 2 (Subsumption) Let C_1, C_2 be two Horn clauses. We say that C_2 *subsumes* C_1 if and only if (iff) C_2 logically entails C_1. In formulae:

$$C_2 \models C_1 \quad \text{(every model of } C_2 \text{ is a model of } C_1\text{)}$$

A proof-theoretic counterpart for Definition 2, based on Robinson's (1965) results on resolution theorem proving, is given in (Bain and Muggleton, 1991).

Definition 3 (θ-subsumption) Let C_1, C_2 be two Horn clauses. We say that C_2 *θ-subsumes* C_1 iff there exists a substitution σ such that $C_2\sigma \subseteq C_1$.
Without loss of generality, we always assume that $dom(\sigma) \subseteq vars(C_2)$.

It is worthwhile to note that θ-subsumption is a strictly weaker order relation than subsumption for first-order logic (Plotkin, 1970; Helft, 1987; Niblett, 1988), as shown by Example 1.

Example 1. (Helft, 1987)
Let C = {P(x), P(f(x))} and D = {P(A), P(f(f(A)))} be two clauses.
C subsumes *D* but *C* does not θ-subsumes *D*.

Nevertheless, contrarily to what stated in (Helft, 1987), definitions 2 and 3 do not become equivalent when we restrict to function-free (but not necessarily constant-free) first order logic, i.e. a first order theory with no functional symbols other than constants, as the following example[3] points out.

Example 2.
Consider the following clauses:
C = {ancestor(Pasquale, Giovanni), ¬parent(Pasquale, Caterina), ¬ancestor(Caterina, Giovanni)}
D = {ancestor(Pasquale, Giovanni), ¬parent(Pasquale, Caterina), ¬parent(Caterina, Maddalena), ¬ancestor(Maddalena, Giovanni)}
E = {ancestor(x, y), ¬parent(x, z), ¬ancestor(z, y)}
E subsumes both *C* and *D* but it θ-subsumes only *C*.

Henceforth, we will adopt θ-subsumption as our model of generalization. Therefore, we extend the same order relation in (Plotkin, 1970), defined in terms of *θ-subsumption*, to the augmented set $\mathfrak{H}$ of the Horn clauses plus the empty program.

Definition 4 (Relative generality) Let C_1, C_2 be two Horn clauses. We say that C_2 is *more general than or equal to* C_1 - or that C_1 is *more specific than or equal to* C_2 - and we write[4] $C_1 \leq C_2$ if and only if (iff) C_1 is the empty program or C_2 *θ-subsumes* C_1, i.e.

$$C_1 \leq C_2 \text{ iff } C_1 \equiv \square \text{ or } \exists\, \sigma : C_2\sigma \subseteq C_1$$

When the condition above holds, we say that C_2 is a *generalization* of C_1 and C_1 is a *specialization* of C_2.

We write $C_1 < C_2$ when $C_1 \leq C_2$ and not($C_2 \leq C_1$) and we say that C_2 is *more general than* C_1 (C_2 is *a proper generalization of* C_1) or C_1 is *more specific than* C_2 (C_1 is *a proper specialization of* C_2).
We write $C_1 \sim C_2$ when $C_1 \leq C_2$ and $C_2 \leq C_1$ and we say that C_1 is *equivalent to* C_2.

θ-subsumption induces a *quasi-ordering* upon the set of Horn clauses, that is, ≤ is reflexive and transitive, but not antisymmetric.

~ is the equivalence relation induced by ≤. Notice that it does not coincide with set equality. Indeed, two equivalent clauses under ~ can be not only alphabetic variants, but even have a different number of literals, as for {P(x), P(f(y))} and {P(f(z))}. Thus, as Plotkin (1970) pointed out, there is a *reduced member* of any equivalence class under ~ and this member is unique to within an alphabetic variant.

Definition 5 (Factor) Let $C = \{l_1, l_2, \ldots, l_n\}$ be a Horn clause. If a subset of the literals in *C* has an MGU σ, the clause *D* obtained by applying σ to *C* is called a *factor* of *C*.

$$D = C\sigma, \text{ where } \sigma = MGU(l_{i1}, l_{i2}, \ldots, l_{ik}),\ k \leq n \text{ and } l_{ij} \in C \text{ for each } j=1,2,\ldots,k$$

Definition 6 (Reduced clause) A clause *C* is *reduced* if it is not equivalent to any proper subset of itself. Formally, *C* is *reduced* iff $D \subseteq C$, $D \sim C$ implies that $C = D$.

Plotkin (1970) reports an algorithm that returns a reduced clause *D*, given any clause *C*. The literals in $C \setminus D$ are called *redundant*.

Example 3. Consider the following clause

$$C = \{\text{bicycle}(x), \neg\text{wheel}(x,y), \neg\text{wheel}(x,z)\}$$

It is easy to see that *C* is not reduced. Indeed, the two literals ¬ *wheel(x,y)*, ¬ *wheel(x,z)* can be unified through the substitution $\sigma = \{ z \leftarrow y \}$ and the resulting factor

$$C' = C\sigma = \{\text{bicycle}(x), \neg\text{wheel}(x,y)\}$$

is logically equivalent to *C* and represents the reduced member of an equivalence class of Horn clauses.

Henceforth, we will always work on the quotient set $\mathfrak{H}/\sim$ and, when convenient, we will denote with

3. Example 2 was suggested by Peter Idestam-Almquist.
4. Differently from Plotkin, we write $C_1 \leq C_2$ to indicate that C_2 is more general than C_1.

the name of a clause the equivalence class it belongs to. Indeed, the definition of ≤ straightforwardly extends to equivalence classes under ~ as follows:

Given two clauses C_1 and C_2, let $[C_1]_\sim$ and $[C_2]_\sim$ denote the corresponding equivalence classes under ~. We say that $[C_1]_\sim \leq [C_2]_\sim$ iff $C_1 \leq C_2$.

3.2. The lattice of Horn clauses

An analysis of the algebraic properties of the ordering relation defined by θ-subsumption led Plotkin to state that the quasi-ordered set $(\mathfrak{C}/\sim, \leq)$ forms a lattice:

$$(\mathfrak{C}/\sim, \vee_{\mathfrak{C}}, \wedge_{\mathfrak{C}})$$

where the lattice operations are defined as follows:

$$[C_1]_\sim \vee_{\mathfrak{C}} [C_2]_\sim = [lgg(C_1,C_2)]_\sim$$
$$[C_1]_\sim \wedge_{\mathfrak{C}} [C_2]_\sim = [C_1 \cup C_2]_\sim$$

By *lgg*(C_1,C_2), we denote the least general generalization of the clauses C_1 and C_2 under θ-subsumption. Formally, we give the following definition:

Definition 7 (Least general generalization) A *least general generalization* (*lgg*) of two clauses is a generalization which is more specific than or equal to any other such generalization.
Formally, given $C_1, C_2 \in \mathfrak{C}$, C is a lgg of $\{C_1, C_2\}$ if:

1. $C_i \leq C$, i=1,2
2. $\forall D$ such that $C_i \leq D$, i=1,2 : $C \leq D$

$lgg(C_1,C_2) = \{ C \mid C_i \leq C$, i=1,2 and $\forall D$ such that $C_i \leq D$, i=1,2 : $C \leq D \}$

This definition can be straightforwardly extended to any subset of $\mathfrak{C}$. Of course, the lgg of any set of clauses is unique under ~, i.e. $|lgg(C_1,C_2)| = 1$.

Least general generalizations are extremely important in the area of inductive learning since they are the only generalizations that are guaranteed to be *correct* - or *consistent*, according to Mitchell's terminology (1982) - whenever correct generalizations exist (Idestam-Almquist, 1993). Furthermore, by definition, $C_1 \vee_{\mathfrak{C}} C_2 = lgg(C_1,C_2) = lgg(C_2,C_1) = C_2 \vee_{\mathfrak{C}} C_1$ and $C_1 \vee_{\mathfrak{C}} C_2 \ldots \vee_{\mathfrak{C}} C_k = lgg(C_1,C_2,\ldots,C_k) = lgg(C_1, lgg(C_2, \ldots, lgg(C_{k-1}, C_k) \ldots)) = C_1 \vee_{\mathfrak{C}} (C_2 \vee_{\mathfrak{C}} (\ldots (C_{k-1} \vee_{\mathfrak{C}} C_k)))$, for any *k*; thus the results produced by any incremental (*data-driven*) learning algorithm are the same as those produced by any batch-learning algorithm, whatever were the order of the training examples, provided that both of them generate the lgg of the given examples. Of course, while a batch learner may perform a search for a lgg of a set of training examples either in a generalization or in a specialization hierarchy, an incremental learning system must search in both of them according to the fact that the new training example is a positive or a negative one, if we do not want to lose the *evidence* provided by one of these two kind of examples. For these *non-monotonic* learning systems, it is relevant to define the concept of *most general specialization* (*mgs*) of a clause against another clause (usually a negative example). A formal definition of mgs can be found in (Esposito *et al.*, 1992; Esposito *et al.*, 1993b; Esposito *et al.*, 1993c).

Example 4.
Consider again the clauses in Example 2:
C = {ancestor(Pasquale, Giovanni), ¬parent(Pasquale, Caterina), ¬ancestor(Caterina, Giovanni)}
D = {ancestor(Pasquale, Giovanni), ¬parent(Pasquale, Caterina), ¬parent(Caterina, Maddalena), ¬ancestor(Maddalena, Giovanni)}
E = {ancestor(x, y), ¬ parent(x, z), ¬ ancestor(z, y)}

$$\text{lgg}(C, D) = F = \{\text{ancestor}(x, y), \neg\text{parent}(x, z), \neg\text{parent}(u, v), \neg\text{ancestor}(v, y)\}$$

Thus, F θ-subsumes both C and D, but, as shown by Example 2, E subsumes both C and D, thus it is less general than F *under subsumption*. Therefore, F is an overgeneralization of C and D.

Example 4 points out that a (strictly) weaker order relation gives rise to a (strictly) weaker model of generalization. Indeed, the example shows clearly that generalization under θ-subsumption is incomplete with respect to subsumption - in particular, θ-subsumption turns out to be incomplete for recursive clauses (Plotkin, 1971b). As told above, we adopt θ-subsumption as our model of generalization. This choice is common to most of the inductive learning systems that learn Horn clauses, such as CIGOL (Muggleton and Buntine, 1988), ML-SMART (Bergadano *et al.*, 1988), LFP2 (Wirth, 1989), IRES (Rouveirol and Puget, 1990), GOLEM (Muggleton and Feng, 1990), OGUST (Vrain, 1990), CLINT (De Raedt, 1991), ITOU (Rouveirol, 1992), and is due to the fact that θ-subsumption is more mechanizable and manageable than subsumption.

Most of the work in the area of concept learning based on Horn clauses casts the generalization problem as a search problem according to (Mitchell, 1982), and refers to the algebraic structure $(\mathfrak{H}/\sim, \vee_{\mathfrak{C}}, \wedge_{\mathfrak{C}})$ to define the search space (hypothesis space). Unfortunately, the properties of the lattice $(\mathfrak{C}/\sim, \vee_{\mathfrak{C}}, \wedge_{\mathfrak{C}})$ are not preserved when the proper subset $\mathfrak{H}/\sim$ is considered. Indeed, Plotkin defined the glb of two (variable disjoint) clauses as their union. The problem with this operator is that, when we restrict to Horn clauses, their union is not always a Horn clause. In fact, if both of them have a head, their union has more than one positive literal. Thus, $\mathfrak{H}/\sim$ *is not a sublattice of* $\mathfrak{C}/\sim$, that is $(\mathfrak{H}/\sim, \vee_{\mathfrak{C}}, \wedge_{\mathfrak{C}})$ *is not a lattice*. On the other hand, if the two clauses had the same head, there would be no problem, since we consider clauses as sets. Observe that, if $C_1 \leq C_2$, that is, $C_2\sigma \subseteq C_1$ for some σ, and C_2 has a head, say H_2, then C_1 also has a head, say H_1. Furthermore, the following condition holds: $H_2\sigma \equiv H_1$. As a consequence, $\sigma \in \text{MGU}(H_1, H_2)$. This suggests a general approach in which we generate a non trivial glb of two definite[5] clauses only when their heads, say H_1, H_2, are unifiable. In that case, the only positive literal of the glb could be the unification of H_1 and H_2. More precisely, it is their most general unification, given that we are looking for the greatest lower bound. The operator defined below follows this reasoning.

Proposition 1.

Every non-empty, finite set of Horn clauses has a most general specialization.

In formulae, if $C_1, C_2 \in \mathfrak{H}/\sim$, then:

$$C_1 \wedge_{\mathfrak{H}} C_2 = \begin{cases} (C_1 \cup C_2)\mu & \text{when } \exists \mu \in \text{MGU}(H_1, H_2) \\ C_1 \cup C_2 & \text{when either } H_1 \text{ or } H_2 \text{ is absent} \\ \square & \text{otherwise} \end{cases}$$

Proof:

1. Assume $\mu \in \text{MGU}(H_1, H_2)$. Let $C = (C_1 \cup C_2)\mu$. Then $C \leq C_i$ for $i = 1, 2$.
   In fact $\exists \mu$ s.t. $C_i\,\mu \subseteq C_1\mu \cup C_2\mu = (C_1 \cup C_2)\mu = C$.
   Assume there exists a clause $\underline{C}$ such that $C \leq \underline{C} \leq C_i$, $i = 1, 2$. $\underline{C}$ is any lower bound of $\{C_1, C_2\}$.
   It suffices to prove that $\underline{C} \sim C$, i.e. that the diagram of Figure 1 can always be filled in along the dotted line, so that it becomes commutative in a unique way.
   Observe that $\underline{C}$ has a head, say $\underline{H}$.
   $C \leq \underline{C}$ implies that $\exists \sigma : \underline{C}\sigma \subseteq C$ and $\underline{H}\sigma \equiv H$ (1)
   $\underline{C} \leq C_i$ implies that $\exists \sigma_i : C_i\,\sigma_i \subseteq \underline{C}$ and $H_i\,\sigma_i \equiv \underline{H}$ (2)

5. Recall that a definite clause is a Horn clause with exactly one positive literal.

Notice that, for i=1, 2, $vars(H_i) \cap dom(\sigma_{3-i}) = \varnothing$ as C_1, C_2 are variable disjoint. Analogously and without loss of generality, $\underline{C}$ can be chosen so that it is variable disjoint with both C_i and thus, for i=1,2, $vars(\underline{H}) \cap dom(\sigma_i) = \varnothing$. By (1), this implies that $H_1 \sigma_1 \sigma_2 \equiv H_2 \sigma_1 \sigma_2 \equiv \underline{H}$.

So $\underline{H}$ is a common instance of H_1, H_2 and is more general, by (1), of their most general common instance H[6]. This means that $\underline{H} \sim H$ and so $\sigma_1 \sigma_2 \in MGU(H_1, H_2)$.

Now, for i=1,2, $H_i \sigma_1 \sigma_2 \sigma \equiv \underline{H}\sigma \equiv H \equiv H_i \mu$ implies that $\sigma_1 \sigma_2 \sigma \equiv \mu$. Moreover, by corollary 9 of (Lassez *et al.*, 1988), σ is invertible.

By (2) again, $C_i \sigma_i \subseteq \underline{C}$ implies that $C_i \sigma_1 \sigma_2 \subseteq \underline{C}$, for i=1, 2, since $\underline{C}$, C_i are disjoint. This implies that $(C_1 \cup C_2)\sigma_1 \sigma_2 \subseteq \underline{C}$.

Therefore, by (1), $(C_1 \cup C_2) \sigma_1 \sigma_2 \sigma \subseteq \underline{C}\sigma \subseteq C = (C_1 \cup C_2) \mu$ which implies that $\underline{C}\sigma = C$, since $\mu = \sigma_1 \sigma_2 \sigma$.

In conclusion, $\underline{C}\sigma = C$, $\underline{H}\sigma = H$ and σ invertible imply that $\exists \sigma^{-1} : C\sigma^{-1} \subseteq \underline{C}$. Hence, by definition, $\underline{C} \leq C$ and so $\underline{C} \sim C$.

2. Let $C = (C_1 \cup C_2)$. Obviously $C \leq C_i$, i=1, 2. Assume that $\exists\, \underline{C} : C \leq \underline{C} \leq C_i$, i = 1, 2. Then, analogously to 1., for i=1, 2, $\exists \sigma_i : C_i \sigma_i \subseteq \underline{C}$ and $C_i \sigma_1 \sigma_2 \subseteq \underline{C}$. It follows that there exists $\sigma_1 \sigma_2$ such that $C \sigma_1 \sigma_2 = (C_1 \cup C_2) \sigma_1 \sigma_2 \subseteq \underline{C}$. Hence, by definition, $\underline{C} \leq C$ and so $\underline{C} \sim C$.
3. Assume $MGU(H_1, H_2) = \varnothing$ and let $C = \square$. By definition of $\leq$, $C \leq C_i$, i = 1, 2. Ad absurdum, assume that $\exists\, \underline{C} : C \leq \underline{C} \leq C_i$, i = 1, 2. Then, as in 1., $\underline{H}$ is a common instance of H_1, H_2. But this contradicts the fact that H_1, H_2 are not unifiable because of the hypothesis. Q.E.D.

The relevance of the operator defined by Proposition 1 in the area of inductive learning lies in the fact that it allows us to adopt a weaker model of generalization in an inductive learning system without losing the possibility of generating (some or all of the) clauses typical of a stronger model of generalization, under certain constraints (see Section 3.4). In fact, a weaker model of generalization is more mechanizable and manageable, but less powerful. Moreover, the operator $\wedge_{\mathfrak{H}}$ is important in different areas other than inductive learning.

In the area of logic programming, (Brogi *et al.*, 1990), independently of us, introduced a similar operator with the purpose of composing logic theories. They provided both an operational (transformational and interpretive) and a model theoretic and a denotational characterization of it and proved that they are equivalent. Unfortunately, their work lies completely outside the area of inductive learning since their dual operator for the union of logic theories is completely static when viewed at the knowledge level (Dietterich, 1988). Furthermore, $(\mathfrak{H}/\sim, \vee_{\mathfrak{H}}, \wedge_{\mathfrak{H}})$ is not an algebra (see Section 3.3.) while an algebra of logic programs has been already defined (Mancarella and Pedreschi, 1988).

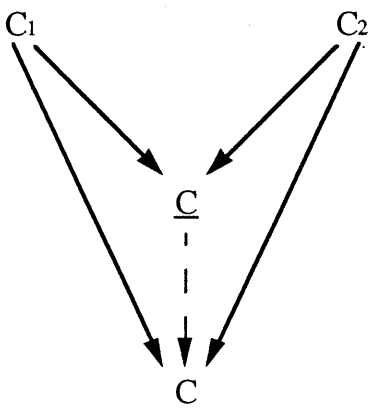


Figure 1. C is the only most general specialization of the clauses C_1 and C_2 in $(\mathfrak{H}/\sim, \vee_{\mathfrak{H}}, \wedge_{\mathfrak{H}})$.

6. Recall that $H \equiv H_i \mu$ where $\mu \in MGU(H_1, H_2)$.

In the area of frame languages or object-centered representations and in the related area of the object-oriented databases and languages, the user is required to hand code a concept hierarchy - a hierarchy of *classes of objects* in the object-oriented terminology. Such concept hierarchies often appear as trees, but generally they can be lattices, sometimes called *tangled hierarchies* (Shapiro *et al.*, 1988). In these structures, a concept is represented by an object or frame or unit, which consists of a list of slots - that represent the properties of the concept - and slot values or terminals. Subsumption is used to organize concept hierarchies. In this area, subsumption can be interpreted either *extensionally*, as set inclusion (containment) (Levesque and Brachman, 1987), or *intensionally* . This last interpretation corresponds to θ-subsumption for some authors (Finin, 1986; Stephens, 1990; Woods, 1991) and to subsumption for others (MacGregor, 1988). Inductive learning can be used to provide a concept with an approximation of its *intension* (Sowa, 1984), by exploiting the instances of the concept as training examples. The added value that inductive learning can give to this area is to provide both knowledge bases and databases with an automatic facility that suggests the class each new added instance belongs to. Moreover, often a concept hierarchy must be updated to insert a new concept. In that case, it is desirable to locate properly the new concept in the hierarchy and to provide it with an intensional definition too. This can be efficiently achieved by applying the operator $\wedge_{\mathfrak{H}}$ to the intensional definitions of the superconcepts of the new concepts without restarting a learning process, which is usually time-consuming. Subsumption techniques offer a unifying framework in the context of object-based representations (Napoli and Ducournau, 1992), since both inheritance and composition are ordering relations weaker than subsumption. Thus the operator $\wedge_{\mathfrak{H}}$ can be also used to solve problems with multiple inheritance, that are currently coped with interactively by asking the user to perform a choice (KEE User's Manual, 1986).

For what concerns the computation of the *least upper bound* (*lub*), Plotkin's algorithm to find the least general generalization (lgg) of any two clauses implements a join operator which is already closed for Horn clauses. Indeed, given two Horn clauses, there exists at most one *selection*[7] that involves the positive literals in the clauses. We only need to extend trivially this algorithm so that the generalization of $\square$ and any clause C is C itself. Let us call $\vee_{\mathfrak{H}}$ the operator corresponding to the extended algorithm. Then every pair of elements C_1, C_2 in the quasi-ordered set $\mathfrak{H}/\sim$ has a *join* $C_1 \vee_{\mathfrak{H}} C_2$ and a *meet* $C_1 \wedge_{\mathfrak{H}} C_2$. It follows that $(\mathfrak{H}/\sim, \vee_{\mathfrak{H}}, \wedge_{\mathfrak{H}})$ - or equivalently $(\mathfrak{H}/\sim, \leq)$ - *is a lattice.*

3.3. Characteristics of $(\mathfrak{H}/\sim, \vee_{\mathfrak{H}}, \wedge_{\mathfrak{H}})$

From the definition of $\vee_{\mathfrak{H}}$ and $\wedge_{\mathfrak{H}}$, it follows that every subset X of $\mathfrak{H}/\sim$ always has both a lub, which is at most the *empty clause* $\varnothing$, and a glb, which is at least the *empty program* $\square$. Therefore $\mathfrak{H}/\sim$ *is a complete lattice.*

$\mathfrak{H}/\sim$ *is not distributive.* In fact, consider the clauses $\{P\}$, $\{\neg Q\}$, $\{P, \neg Q\}$.
$\{P,\neg Q\} \vee_{\mathfrak{H}} (\{P\} \wedge_{\mathfrak{H}} \{\neg Q\})$ is $\{P,\neg Q\}$ while $(\{P,\neg Q\} \vee_{\mathfrak{H}} \{P\}) \wedge_{\mathfrak{H}} (\{P,\neg Q\} \vee_{\mathfrak{H}} \{\neg Q\})$ is $\square$. An immediate consequence of this result is that $(\mathfrak{H}/\sim, \vee_{\mathfrak{H}}, \wedge_{\mathfrak{H}})$ *is not an algebra.*

$\mathfrak{H}/\sim$ *contains ascending and descending chains of infinite length.*

An example of infinite strictly ascending chain bounded above by $Q(x_1) \leftarrow \varnothing$ is $\{C_i\}_{i\geq 0}$ where[8]

$$C_0 = \{ Q(x_1), \neg P(x_1, x_1) \}$$

$$C_{i+1} = (C_i - \{ \neg p(x_{2^i}, x_1) \}) \cup \bigcup_{j=2^i}^{2^{i+1}-1} (\{ \neg p(x_j, x_{j+1}) \}) \cup \{ \neg p(x_{2^{i+1}}, x_1) \}$$

7. Given two clauses, a *selection* is a pair of literals, taken one from each clause, having the same predicate symbol and sign.
8. For clarity sake, in this example we use non variable disjoint clauses.

The body of C_{i+1} contains a number of literals equal to twice as many literals in the body of C_i.

An example of infinite strictly descending chain, adapted from (Plotkin, 1970), bounded below by $D_0 = \{ P(x, x), \neg P(x, x) \}$, is $\{D_i\}_{i \geq 1}$ where

$D_1 = \{ P(x_1, x_2), \neg P(x_1, x_2) \}$

$D_i = D_{i-1} \cup \{ \neg P(x_i, x_{i+1}) \}$

It is worthwhile to note that an immediate consequence of the existence of infinite both descending and ascending chains is that the search space of any inductive system that learns Horn clauses is inherently infinite, when further language restrictions are not considered. Even more so, both general-to-specific and specific-to-general search strategies may go through an infinite path. This consideration points out the relevance of a proper definition both of the goal of the search - for instance, least general generalization vs. most general generalization - and of the language biases which, together with the adopted model of generalization, may reduce/increase the size of the search space, hopefully without reducing its scope (or at least reducing the scope of the search space only on meaningless clauses).

The effect of different language biases on the algebraic structure of the search space and on its size and scope is investigated in the next section.

3.4. The effect of language biases on the algebraic structure of the search space

In this section, we consider four distinct language biases, namely *domain-restriction*, *range-restriction* (De Raedt, 1991), *linkedness* and *object identity*. The first three biases constrain the kind of Horn clauses allowed, the last one limits the kind of substitutions considered. Subsequently, we define formally domain-restriction, range-restriction and object identity. As to linkedness, we refer to Definition 1 in Section 2.1.

Definition 8 (Domain-restriction) A clause C is a *domain-restricted* clause iff all variables occurring in the body of the clause also occur in the head.

A domain-restricted clause does not contain local variables, i.e. variables which occur in the body of the clause but not in its head. Domain-restricted clauses have some desirable properties as to the problems of non monotonic reasoning (Barbuti *et al.*, 1990) and of reducing the computational costs of the search performed by FOIL and FOCL. Indeed, the first version of FOCL adopted an iterative widening search strategy when generating literals to test. This means that FOCL first attempted to learn a domain-restricted clause and then, if such a search failed, it tried to learn linked clauses.

Domain-restricted clauses should not be confused with range-restricted clauses.

Definition 9 (Range-restriction) A clause C is *range-restricted* iff all variables occurring in the head of the clause also occur in the body.

Range-restricted function-free clauses are also called *groundable* (Helft, 1989).

It is worthwhile to note that domain-restricted clauses are also linked clauses, while this relation does not hold for range-restricted clauses.

Example 5.

Consider the following clauses:

$C = \{\text{bicycle}(x,y,z), \neg\text{wheel}(x,y)\}$

$D = \{\text{bicycle}(x), \neg\text{red}(x), \neg\text{large}(y)\}$

$E = \{\text{bicycle}(x), \neg\text{wheel}(x,y)\}$

C is a domain-restricted clause, thus it is also linked, but it is not range-restricted, because of *z*. *D* is range-restricted, but it is not linked and it is not a domain-restricted clause, since variable *y* is local to the last literal in the body of the clause. *E* is linked and range-restricted, but it has a local variable, too.

Definition 10 (Object identity) Variables with different names denote distinct objects (or *units*).

Object identity is a restriction on the kind of substitutions allowed. More precisely, substitutions are necessarily injective functions under such an assumption. This means that only one-to-one variable bindings are allowed, instead of many-to-one variable bindings. Formally, object identity corresponds to an equality theory with an axiom schema of the form $f(t_1,t_2,\ldots,t_n)\neq g(t'_1,t'_2,\ldots,t'_m)$ whenever $f \neq g$, where t_k and t'_k, k=1,2,..., denote the terms of the language.

For instance, the following clause:

arch(x, y, z) ← on-top(x, y), left-of(y, z)

is equivalent to:

arch(x, y, z) ← on-top(x, y), left-of(y, z), [x≠y], [x≠z], [y≠z]

when object identity holds.

Object identity assumption corresponds to introduce *constraints* (Jaffar and Lassez, 1987) in form of *difference links* of the type *[x≠y]* into the Horn clause representation of inductively learned rules. Therefore, under this assumption, we should speak more precisely of Inductive Constraint Logic Programming (ICLP).

Definition 11 (θ-subsumption under object identity) Let C_1, C_2 be two Horn clauses. We say that C_2 *θ-subsumes* C_1 *under object identity* (C_2 θ_{OI}*-subsumes* C_1) iff there exists a substitution σ such that $C_2\sigma \subseteq C_1$ and σ *is injective.*

Definition 12 (Relative generality under object identity) Let C_1, C_2 be two Horn clauses. We say that C_2 is *more general than or equal to* C_1 *under object identity* and we write $C_1 \leq_{OI} C_2$ iff C_1 is the empty program or C_2 θ_{OI}-subsumes C_1, that is,

$$C_1 \leq_{OI} C_2 \text{ iff } C_1 \equiv \square \text{ or } \exists\, \sigma : C_2\sigma \subseteq C_1 \text{ and } \sigma \text{ is injective}$$

Similarly to Definition 4, we can define the concepts of *(proper) generalization/specialization* and *equivalence* under object identity. The corresponding symbols will be endowed with the subscript OI. The following proposition holds:

Proposition 2.
Let C_1, C_2 be two Horn clauses, then:

$$C_1 \leq_{OI} C_2 \Rightarrow C_1 \leq C_2$$

while the opposite is not true in general.

Proof:
Trivial.

Therefore, θ-subsumption under object identity is a model of generalization weaker than θ-subsumption, which in turn is weaker than subsumption.

Example 6.
Consider again the clause of Example 3:

C = {bicycle(x), ¬wheel(x,y), ¬wheel(x,z)}

It is easy to see that under object identity *C* cannot be further reduced. Indeed, the two clauses *C* and C' = {bicycle(x), ¬wheel(x,y)} define two distinct equivalence classes under $\sim_{OI}$.

Example 7.

a) Consider the following Horn clauses:

$C_0 = \{ Q(x_1), \neg P(x_1, x_1) \}$

$C_1 = \{ Q(x_1), \neg P(x_1, x_2), \neg P(x_2, x_1) \}$

$C_2 = \{ Q(x_1), \neg P(x_1, x_2), \neg P(x_2, x_3), \neg P(x_3, x_4), \neg P(x_4, x_1) \}$

$C_0 \leq C_1 \leq C_2$ but not $(C_0 \leq_{OI} C_1 \leq_{OI} C_2)$.

Note that C_0, C_1 and C_2 are nodes of the infinite strictly ascending chain in $(\mathfrak{H}/\sim, \leq)$ cited in Section 3.3. In $(\mathfrak{H}/\sim, \leq_{OI})$ such an infinite chain does no longer exist, even though all the clauses C_i still exist.

b) Consider the following Horn clauses:

$C_1 = \{ P(x_1, x_2), \neg P(x_1, x_3), \neg P(x_3, x_2) \}$

$C_2 = \{ P(x_1, x_2), \neg P(x_1, x_3), \neg P(x_3, x_4), \neg P(x_5, x_2) \}$

$C_1 \leq C_2$ but not $(C_1 \leq_{OI} C_2)$.

Example 8.

Consider the following Horn clauses:

$D_0 = \{ P(x, x), \neg P(x, x) \}$

$D_1 = \{ P(x_1, x_2), \neg P(x_1, x_2) \}$

$D_2 = \{ P(x_1, x_2), \neg P(x_1, x_2), \neg P(x_2, x_3) \}$

$D_{i+1} \leq D_i$ and $D_{i+1} \leq_{OI} D_i$, but while the condition $D_0 \leq D_i$ holds for i=1,2,..., the condition $D_0 \leq_{OI} D_i$ is false for any i=1,2,...

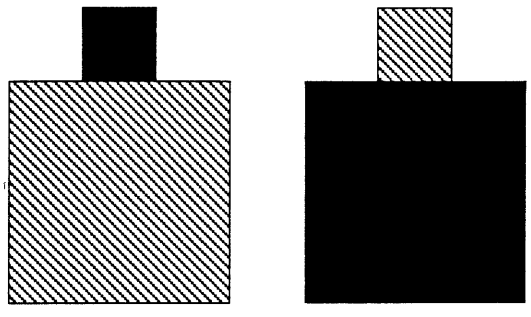
Note that D_0, D_1 and D_2 are nodes of the infinite strictly descending chain in $(\mathfrak{H}/\sim, \leq)$ cited in Section 3.3. In $(\mathfrak{H}/\sim, \leq_{OI})$ such an infinite chain exists but D_0 does no longer belong to it. Moreover, the risk that a (general-to-specific) search into a specialization hierarchy goes infinitely through this path is purely theoretical and belongs to the sphere of the asymptotic analysis of the behaviour of learning systems, since it can happen only in the unrealistic case one of the training examples consists of an infinite number of literals.

Example 9. (Manago and Kodratoff, 1987)

Let E_1 and E_2 be two positive examples of the predicate *blocks(x)* (see Figure 2):

blocks(obj1) :- part-of(obj1, p1), part-of(obj1, p2), on(p1, p2),
is-a(p1, cube), is-a(p2, cube),
size(p1, small), size(p2, big),
color(p1, black), color(p2, stripes)

blocks(obj2) :- part-of(obj2, p3), part-of(obj2, p4), on(p3, p4),
is-a(p3, cube), is-a(p4, cube),
size(p3, small), size(p4, big),
color(p3, stripes), color(p4, black)



E1 E2

Figure 2. Two examples from the blocks-world domain.

The lgg of E_1 and E_2, *lgg*(E_1, E_2), is the following linked Horn clause G:

G blocks(x) :- part-of(x, x1), part-of(x, x2), part-of(x, x3), part-of(x, x4), on(x1, x2), is-a(x1, cube), is-a(x2, cube), is-a(x3, cube), is-a(x4, cube), size(x1, small), size(x2, big), color(x3, black), color(x4, stripes)

which has a number of literals greater than both E_1 and E_2. Indeed, we have: $G\,\sigma_1 \subseteq E_1$ and $G\,\sigma_2 \subseteq E_2$, where $\sigma_1 = \{ x \leftarrow obj1, x_1 \leftarrow p_1, x_2 \leftarrow p_2, x_3 \leftarrow p_1, x_4 \leftarrow p_2 \}$ and $\sigma_2 = \{ x \leftarrow obj2, x_1 \leftarrow p_3, x_2 \leftarrow p_4, x_3 \leftarrow p_4, x_4 \leftarrow p_3 \}$ are many-to-one variable bindings. Under object identity assumption, G is no longer a generalization of E_1 and E_2 since many-to-one variable bindings are not permitted. The following two linked Horn clauses, G_1 and G_2,

G_1 blocks(x) :- part-of(x, x1), part-of(x, x2), on(x1, x2), is-a(x1, cube), is-a(x2, cube), size(x1, small), size(x2, big)

G_2 blocks(x) :- part-of(x, x1), part-of(x, x2), is-a(x1, cube), is-a(x2, cube), color(x1, black), color(x2, stripes)

have the property to be more specific (less general) than any other generalization of E_1 and E_2 under object identity, but their mutual generality cannot be compared. In formulae:

1. $E_i \leq_{OI} G_1$ and $E_i \leq_{OI} G_2$, i = 1,2
2. $\forall D$ such that $E_i \leq_{OI} D$, i=1,2 : not $(D <_{OI} G_j)$, j=1,2

In particular, when $D = G_k$, $k \neq j$, the following conditions hold: not $(G_1 <_{OI} G_2)$ and not $(G_2 <_{OI} G_1)$. Furthermore, both G_1 and G_2 are reduced, thus they are not equivalent, i.e. not($G_1 \sim_{OI} G_2$). It is easy to see that both in $(\mathfrak{H}/\sim, \leq)$ and in $(\mathfrak{H}/\sim, \leq_{OI})$ G exists and is a specialization of both G_1 and G_2, i.e. $G < G_1, G < G_2$ and $G <_{OI} G_1, G <_{OI} G_2$. Nevertheless, as told above, in $(\mathfrak{H}/\sim, \leq_{OI})$, G is no longer a generalization of E_1 and E_2. The example confirms the fact that θ_{OI}-subsumption is a model of generalization weaker than θ-subsumption.

The previous example makes clear the fact that, under object identity assumption, the set of the Horn clauses is no longer a lattice, but it is simply a quasi-ordered set, that is, $(\mathfrak{H}/\sim, \leq)$ is a lattice, while $(\mathfrak{H}/\sim, \leq_{OI})$ is not. Quasi-ordered search spaces that are not lattices are *finitary* as to the problem of finding a lgg of two or more clauses. That is, there exists a set of lgg of any pair of elements in $\mathfrak{H}/\sim$ and this set is at most finite, while lattices are *unitary* by definition (there exists a unique reduced lgg). As a consequence, when the search space of a learning algorithm is finitary, Definition 7 should be modified as follows:

Definition 13 (Lgg in quasi-ordered spaces) In a quasi-ordered space, a *least general generalization* (lgg) of two clauses is a generalization which is not more general than any other such generalization, that is, it is either more specific than or not comparable to any other such generalization.
Formally, given $C_1, C_2 \in \mathfrak{C}$, C is a lgg of $\{C_1, C_2\}$ if:

1. $C_i \leq C$, i=1,2
2. $\forall D$ such that $C_i \leq D$, i=1,2 : not $(D < C)$

$lgg(C_1, C_2) = \{ C \mid C_i \leq C$, i=1,2 and $\forall D$ such that $C_i \leq D$, i=1,2 : not($D < C$) $\}$

Of course, the same results apply to Horn clause logic.

Generalizations in the quasi-ordered space $(\mathfrak{H}/\sim, \leq_{OI})$ are also called *object-oriented* (or *unit-oriented*) generalizations, in contrast with generalizations in $(\mathfrak{H}/\sim, \leq)$, that are said *feature-oriented* generalizations (Manago and Kodratoff, 1987). In the former, variables represent distinct conceptual objects (*units*) and

must be bound to different units, in the latter many-to-one variable bindings are allowed.

$(\mathfrak{H}/\sim, \leq_{OI})$ has another desirable property for the problem of generalization. In this space there exist only finite strictly ascending chains since the set of all generalizations of a clause C corresponds to the power set of C, i.e. the set of all subsets of the literals of C, thus each proper generalization of C has a number of literals less than the number of literals of C (VanLehn, 1989). In formulae:
Let C be a clause and $GEN_{OI}(C)$ the set of all the generalizations of C under object identity. It holds that:

$$GEN_{OI}(C) = \{ D \in \mathfrak{H}/\sim \mid |D| < |C| \} = 2^C$$

where $|A|$ denotes the number of literals in the clause A and $<$ denotes the usual relational operator between integers.

As a consequence, the cardinality of $lgg_{OI}(C_1,C_2)$ in $(\mathfrak{H}/\sim, \leq_{OI})$ is upper bounded by the following condition:

$$|lgg_{OI}(C_1,C_2)| \leq |GEN_{OI}(C_1) \cap GEN_{OI}(C_2)| = 2^n, \text{ where } n = \min \{ |C_1|, |C_2| \}.$$

On the contrary, the feature-oriented generalization G of the two clauses E_1 and E_2 in Example 9 is *longer* than both E_1 and E_2. Indeed, when many-to-one variable bindings are allowed, there exist infinite strictly ascending chains in the lattice of the Horn clauses, as stated in Section 3.3.

Given two clauses C_1 and C_2, it holds that: $lgg(C_1,C_2) \leq D$ for each D in $lgg_{OI}(C_1,C_2)$.

All the results above extend straightforwardly to the proper subset of the linked Horn clauses.

These results show clearly the fact that θ_{OI}-subsumption is a weaker model of generalization than θ-subsumption, that is, the lgg's generated by assuming θ_{OI}-subsumption as model of generalization are overly general w.r.t. the corresponding unique θ-subsumption-based lgg.

The operator defined by Proposition 1 in Section 3.2 is able to recover, under certain constraints, the unique lgg of two clauses that can be generated under θ-subsumption, but not under θ_{OI}-subsumption. For instance, in Example 9, the operator $\wedge_{\mathfrak{H}}$ can be used to generate $G = lgg(E_1, E_2)$ from G_1 and G_2 (after having standardized apart G_1 and G_2) as follows:

G_1 = { blocks(x), ¬part-of(x, x1), ¬part-of(x, x2), ¬on(x1, x2), ¬is-a(x1, cube), ¬is-a(x2, cube), ¬size(x1, small), ¬size(x2, big) }

G_2 = {blocks(y), ¬part-of(y, x3), ¬part-of(y, x4), ¬is-a(x3, cube), ¬is-a(x4, cube), ¬color(x3, black), ¬color(x4, stripes) }

$G = G_1 \wedge_{\mathfrak{H}} G_2 = (G_1 \cup G_2)\mu =$
= { blocks(x), ¬part-of(x, x1), ¬part-of(x, x2), ¬part-of(x, x3), ¬part-of(y, x4), ¬on(x1, x2), ¬is-a(x1, cube), ¬is-a(x2, cube), ¬is-a(x3, cube), ¬is-a(x4, cube), ¬size(x1, small), ¬size(x2, big), ¬color(x3, black), ¬color(x4, stripes) }

where μ = { y ← x }. Figure 3 shows the explored portion of the search space for this example.

It is relevant to observe that the process of searching for the lgg's of any two clauses under the weaker model of generalization represented by θ_{OI}-subsumption and then applying the operator $\wedge_{\mathfrak{H}}$ in order to

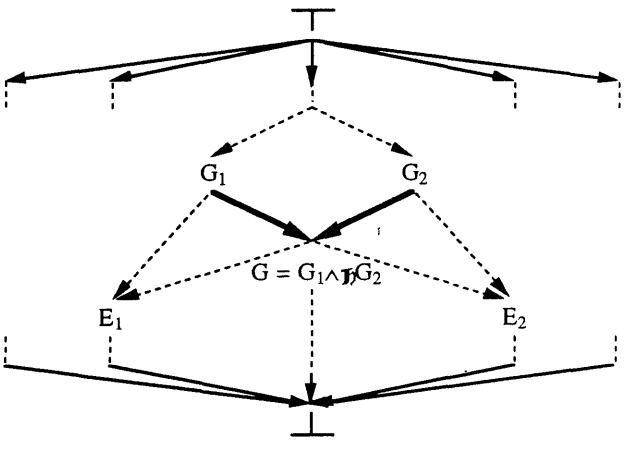


Figure 3. A portion of the lattice $(\mathfrak{H}/\sim, \leq)$. $\top$ and $\bot$ denote the top element (the empty clause *blocks(x) :- ∅*) and the bottom element (the empty program □), respectively.
A node represents an equivalence class under ~. A dashed arrow from a node A to a node B means that A is more general than B. A plain arrow from A to B means that A is more general than B and there is no C such that A is more general than C and C is more general than B.

recover some generalizations of the same clauses under θ-subsumption does not reduce the computational complexity of the problem, since the following theorem due to Haussler (1989) holds:*"The problem of determining if there is an existential conjunctive concept consistent with a sequence of m of examples over an instance space defined by n attributes (where m and n are variable) is NP-complete, even when there are no binary relations defined, each attribute is Boolean valued, and each example contains exactly two objects."*

θ_{OI}-subsumption is the generalization model underlying most of the earliest inductive learning systems developed in the seventies, such as ARCH (Winston, 1970), SPROUTER (Hayes-Roth, 1974), Thoth-pb (Vere, 1977), INDUCE 1.1 (Larson, 1977), and some more recent empirical learning systems, such as MAGGY (Manago and Kodratoff, 1987), INDUBI (Semeraro, 1988; Esposito, 1990) and SIERRA (VanLehn, 1989). A consistency-preserving refinement operator that copes with the dual problem of finding a *most general specialization (mgs)* in the generalization model defined by θ_{OI}-subsumption can be found in (Esposito *et al.*, 1992; Esposito *et al.*, 1993b; Esposito *et al.*, 1993c).

Henceforth, we will denote with $\mathfrak{D}$ the set of the domain-restricted Horn clauses, with $\mathfrak{R}$ the set of the range-restricted Horn clauses and with $\mathfrak{L}$ the set of the linked Horn clauses. Obviously, $\mathfrak{D}/\sim \subset \mathfrak{L}/\sim \subset \mathfrak{H}/\sim \subset \mathfrak{C}/\sim$. As claimed by Buntine (1988), surprisingly generalization and specialization hierarchies possess very different properties. In the following, by *size* of the search space we mean the number of equivalence classes in a specialization/generalization hierarchy. Both domain-restriction and range-restriction and linkedness reduce the size of both specialization and generalization hierarchies. Conversely, object identity increases the size of the hypothesis space, as Examples 6, 7, 8 and 9 show, but it reduces the *scope* of the search, as Example 9 shows. Therefore, differently from some authors (Jung, 1993, pg.89), we claim that a weaker generalization model, like that induced by object identity, does not necessarily reduce the size of the search space. Table I summarizes the most relevant properties of the search spaces defined adopting one of the previously presented biases or a combination of them.

Table I

Legend: *Q.o. set* is an abbreviation for *quasi-ordered set*. *Finite* (*infinite*) generalization/specialization hierarchies means that there exist finite (infinite) chains in the generalization/specializations hierarchies. No results are reported for $(\mathfrak{H}/\sim,\vee_{\mathfrak{C}},\wedge_{\mathfrak{C}})$ since $\mathfrak{H}/\sim$ is not closed w.r.t. the $\wedge_{\mathfrak{C}}$ operator.

Space	Algebraic structure	Generalization hierarchies	Specialization hierarchies
$(\mathfrak{C}/\sim,\vee_{\mathfrak{C}},\wedge_{\mathfrak{C}}) =$ $= (\mathfrak{C}/\sim, \leq)$	Lattice	Infinite	Infinite
$(\mathfrak{H}/\sim,\vee_{\mathfrak{C}},\wedge_{\mathfrak{C}})$	-	-	-
$(\mathfrak{H}/\sim,\vee_{\mathfrak{H}},\wedge_{\mathfrak{H}}) =$ $= (\mathfrak{H}/\sim, \leq)$	Lattice	Infinite	Infinite
$(\mathfrak{H}/\sim, \leq_{OI})$	Q.o. set	Finite	Infinite
$(\mathfrak{L}/\sim,\vee_{\mathfrak{H}},\wedge_{\mathfrak{H}}) =$ $= (\mathfrak{L}/\sim, \leq)$	Lattice	Infinite	Infinite
$(\mathfrak{L}/\sim, \leq_{OI})$	Q.o. set	Finite	Infinite
$(\mathfrak{D}/\sim, \vee_{\mathfrak{H}},\wedge_{\mathfrak{H}}) =$ $= (\mathfrak{D}/\sim, \leq)$	Lattice	Finite	Finite
$(\mathfrak{D}/\sim, \leq_{OI})$	Q.o. set	Finite	Finite
$(\mathfrak{R}/\sim,\vee_{\mathfrak{H}},\wedge_{\mathfrak{H}}) =$ $= (\mathfrak{R}/\sim, \leq)$	Lattice	Infinite	Infinite
$(\mathfrak{R}/\sim, \leq_{OI})$	Q.o. set	Finite	Infinite

4. Experiments with FOIL and FOCL

This section presents some experimental results obtained by running FOIL and FOCL[9] both on classical tasks from the machine learning literature and on real world applications.

Henceforth, unless otherwise stated, we will always refer to the space $\mathfrak{L}/\sim$ of the linked Horn clauses, since this is the search space for both the learning systems FOIL and FOCL. It is worthwhile to note that both FOIL and FOCL adopt θ-subsumption as model of generalization and do not make the object identity assumption, thus the search space for these systems is $(\mathfrak{L}/\sim, \leq)$.

The high level view of both FOIL and FOCL algorithms has two main stages: *separate* and *conquer*. The *separate* stage of the algorithm is basically a loop that checks for the completeness of the current rule and, if this check fails, begins the search for a new consistent clause, while the *conquer* stage performs a general-to-specific search to construct the body of the new clause.

Therefore, as to the search space, we can more precisely state that the search space for the separate stage is $2^{\mathfrak{L}/\sim}$, i.e. the power set of $\mathfrak{L}/\sim$, while the conquer stage searches for a consistent clause in the only specialization hierarchy of $(\mathfrak{L}/\sim, \leq)$ rooted into the linked Horn clause whose head contains the predicate to be learned and whose body is empty, i.e. $P(X) \leftarrow \varnothing$, if P is the predicate that denotes the concept to learn. From Table 1, we know that such a specialization hierarchy contains infinite chains.

In the following, we point out some traps and pitfalls in which both FOIL and FOCL fall and provide an explanation for these negative results on the ground of the revised logical framework in Section 3.

4.1. Learning the concept of an arch

The objects in Figure 4 are the training examples supplied to FOIL and FOCL in order to learn the nature of an arch much like the arch which Winston describes (Winston, 1970). The set $A = \{A_1, A_2\}$ includes examples of arches and the set $N = \{N_1, N_2, N_3\}$ contains examples of objects that are not arches. The predicates used to describe the objects are the following:

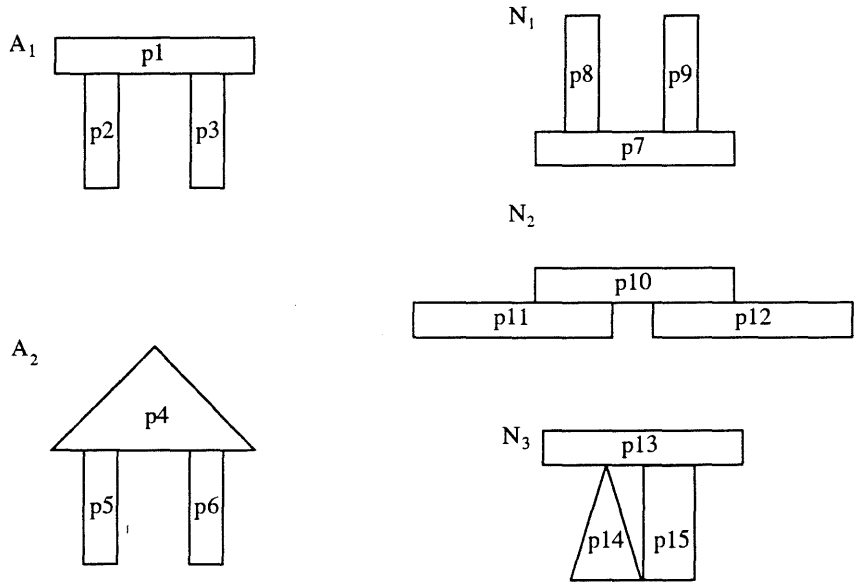


Figure 4. Training set for the problem *arch*.

9. All the experiments were run using FOIL5.0 (Quinlan and Cameron-Jones, 1993) and FOCL-1-2-3 Version 1.1 (Pazzani and Brunk, 1993).

arch(X)	X is an arch
cont(X, Y)	X contains Y
on(X, Y)	X is on Y
touch(X, Y)	X touches Y
or-hor(X)	X is horizontally oriented
or-ver(X)	X is vertically oriented
or-not-appl(X)	X has no orientation
sh-rect(X)	X has a rectangular shape
sh-tri(X)	X has a triangular shape

The complete descriptions of the five examples are reported in Appendix A.

FOIL takes .1 seconds to generate no clause, that is, it is not able to find a definition for the concept *arch* from the given training set.

FOCL too is not able to generate a complete and consistent concept definition and, moreover, the learning process does not converge, that is, FOCL's search does not terminate.

As stated above, the search space of the conquer stage in FOCL is the specialization hierarchy of $(\mathfrak{L}/\sim, \leq)$ rooted into the linked Horn clause $arch(X) \leftarrow \varnothing$. Therefore, one reason for FOCL not to converge towards a solution is that the search goes through an infinite strictly descending chain of the (θ-)subsumption hierarchy, such as $\{D_i\}_{i\geq 1}$ of Example 8, that is an unrealistic hypothesis, as already stated. Another possibility is that the operation of adding a literal to the body of the current clause actually generalizes the clause rather than specializes it. In such a case, the search would go through an infinite strictly ascending chain of the θ-subsumption hierarchy, such as $\{C_i\}_{i\geq 0}$ of Example 7a. But no infinite strictly ascending chain in the search space can be generated by FOCL's *conquer* stage since at each step the new clause strictly contains the previous one, while this condition does not hold for the infinite strictly ascending chains of clauses in $(\mathfrak{L}/\sim, \leq)$. Therefore, this hypothesis is ruled out.

An analysis of the ongoing running of FOCL provides a simple explanation of this apparently contradictory behaviour. Figure 5 is a snapshot of the state of FOCL's search at a time *t*. It shows an AND/

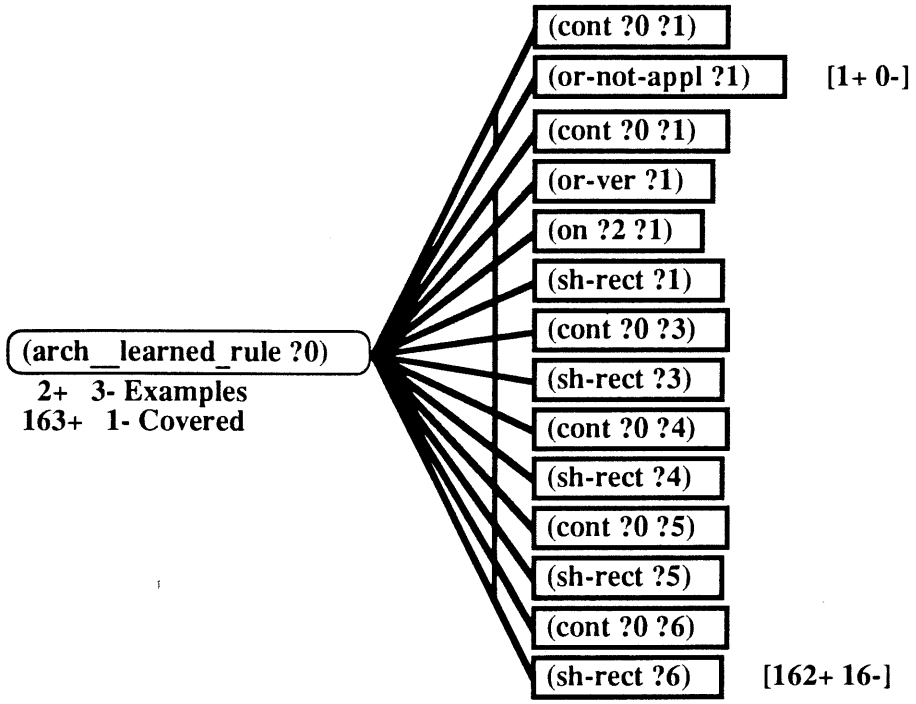


Figure 5. Partial AND/OR graph produced by FOCL for the problem *arch*. Variables are denoted by ?n, n=0,1,2...

OR graph corresponding to the following two clauses:

C_1 arch(?0) :- cont(?0, ?1), or-not-appl(?1)

C_2 arch(?0) :- cont(?0, ?1), or-ver(?1), on(?2, ?1), sh-rect(?1),
cont(?0, ?3), sh-rect(?3),
cont(?0, ?4), sh-rect(?4),
cont(?0, ?5), sh-rect(?5),
cont(?0, ?6), sh-rect(?6),...

The former is a consistent clause, which covers only the positive example A_2. The latter is the clause that FOCL is generating in the attempt to achieve completeness. Of course, actually this clause is a partial one, but it is enough to give us an insight of what is going on:

FOCL's conquer stage searches in the wrong specialization hierarchy

1st trap:

FOCL's conquer stage performs its search in the lattice $(\mathbb{L}, \leq)$ ***of all the linked Horn clauses rather than in the lattice*** $(\mathbb{L}/\sim, \leq)$ ***of all the equivalence classes*** (that have a linked Horn clause as a reduced member).

Indeed, it is easy to see that the second clause in the rule for arch is logically equivalent to the clause:

C_3 arch(?0) :- cont(?0, ?1), or-ver(?1), on(?2, ?1), sh-rect(?1),
cont(?0, ?3), sh-rect(?3)

In formulae:

$$C_2 \sim C_3, \text{ i.e. } C_2 \leq C_3 \text{ and } C_3 \leq C_2$$

Indeed $C_3\sigma_1 \subseteq C_2$ and $C_2\sigma_2 \subseteq C_3$ hold, where $\sigma_1 = \{\}$ and $\sigma_2 = \{ ?4 \leftarrow ?3, ?5 \leftarrow ?3, ?6 \leftarrow ?3 \}$.

$$C_2, C_3 \in [C_2]_\sim$$

FOCL does not implement an algorithm for testing the equivalence of the newly generated hypothesis with respect to the previous one. As a consequence, the search continues by generating-and-testing hypotheses that belong to the same equivalence class as the previously generated hypothesis, actually remaining inside the same node of the specialization hierarchy and with no mean to go out from there.

In other words, the conquer stage in FOCL's high level view defines incorrectly its termination condition, therefore there is no guarantee of termination.

In the running above, we introduced the relational cliché *CONT* by which FOCL is said to test the couples of literals:

$$\text{cont}(X,Y),\ Q(V_1, V_2, \ldots, V_t)$$

In fact, the pair of literals *cont(?0, ?n), sh-rect(?n)* turns out to have the maximum information gain at each step of the search for a new literal to add. We chose to use relational clichés since a previous experiment with no relational cliché pointed out that FOCL could not generate any concept definition, exactly as FOIL. Moreover, setting one of FOCL's built-in functions, namely the *eql* built-in function that allows it to test also the literals *eql(X, Y)* (and their negations)[10], had no effect on the result since these literals have a low information gain.

It is interesting to observe that Quinlan (1990) reports a result obtained by FOIL on the task of learning the concept of arch[11]. If we compare the experiment performed by Quinlan to the similar experiment presented above, we can easily observe that Quinlan's formulation of the arch problem makes the learning process easier. In fact, FOIL learns the predicate *arch(A,B,C)* instead of *arch(A)*. This is apparently a slight difference, but as a matter of fact it does not allow the learning system to exploit any training

10. This would allow FOCL to create literals corresponding to *difference links.*

11. FOIL5.0 is released with some data files. The file containing the input data for the *arch* problem is *winston.d.*

instance whose number of parts is different from three. For instance, FOIL cannot exploit training examples like those in Figure 6, in which a post consists of two bricks (A_3) or three posts support a lintel (A_4). This represents a strong limitation when a learning system has to cope with real world domains. Indeed in the field of computer vision - the area addressed by Winston's thesis (we recall that Winston refers to instances with the name of *scenes*) - the processes of segmentation and edge detection are affected by noise, thus it happens very rarely that all the observed objects have exactly the same number of components. Moreover, another cause of this is the intrinsic variability of the instances of a class. In the area of structural pattern recognition, local deformations can cause a primitive in a pattern to be split into two, or two separate ones to be merged into one (Sanfeliu and Fu, 1983). For example, in character recognition, a character stroke may be broken into two segments after preprocessing or two close segments may be joined into one due to noise (Wong and You, 1985). An example of a real world domain concerning the understanding of 2-dimensional scenes is reported in Section 4.4. In this domain, the problem addressed above is known with the name of *fragmentation* and strongly affects the phase of *layout analysis*.

As a matter of fact, our formulation of the *arch* problem is more similar to that in (Larson, 1977) and requires that the learning system be able to determine autonomously the number of components (and their characteristics) that are sufficient to discriminate the positive instances of the target concept from the negative ones. Conversely, the representation adopted by Quinlan biases the learning system towards the generation of concept definitions whose clauses are both domain-restricted and range-restricted, since all the arguments of the predicates in the language of the problem refer to *components*, with the exception of the first argument of the predicate *cont*, that refers to the arch itself.

4.2. Learning the concept of a bicycle

Helft (1987) reports a very simple learning task. The learning system is given two descriptions of a bicycle:

B_1: bicycle(obj1) :- wheel(obj1, p1), wheel(obj1, p2)
B_2: bicycle(obj2) :- wheel(obj2, p3), wheel(obj2, p4)

Helft observes that the best generalization á la Plotkin obtainable from these examples is:

G: bicycle(X) :- wheel(X, Y)

since the clause:

G': bicycle(X) :- wheel(X, Y), wheel(X, Z)

is logically equivalent to *G* (*G* is a *factor* of *G'*). As a consequence, the number of wheels does not appear in the concept description of bicycle, even though it is a relevant characteristic.

As expected, both FOIL and FOCL learn the same rule *bicycle(X)* ← ∅, when cope with this task. This

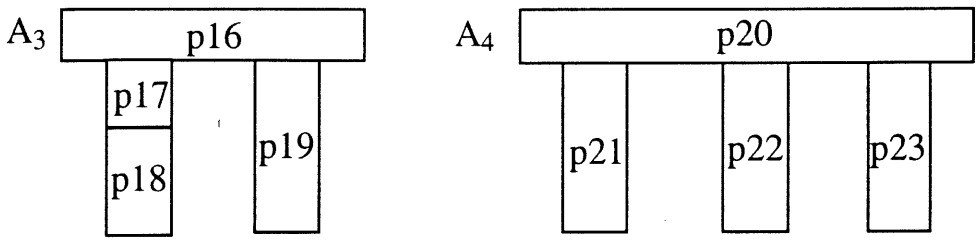


Figure 6. *Four-component* instances for the problem *arch*.

result is easily explained by the fact that these two systems successfully adopt a *lazy* learning strategy that biases the search towards the generation of maximally general discriminant descriptions rather than least general generalizations. Therefore, when no negative examples are provided, the rule discovered will always consist of a single clause whose body is empty.

The behaviour of these two systems is more interesting on two slightly different learning tasks, that we created purposely. The two learning problems differ from each other by the description of a unique negative example that we added to the original training set.

In the former problem, the training set consists of the positive examples B_1 and B_2 and the negative example N_1:

N_1: non-bicycle(obj3) :- $\varnothing$

Both FOIL and FOCL generate the same clause:

B: bicycle(A) :- wheel(A, B) (bicycle_learned_rule ?0) — (wheel ?0 ?1) [4+ 0-]

In the latter problem, the only training example, N_2, is represented by the following clause:

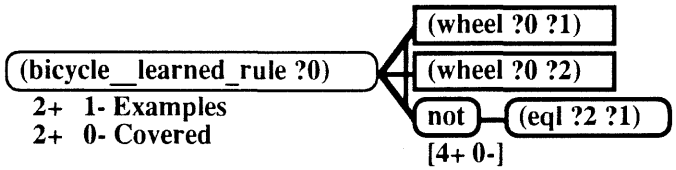
N_2: non-bicycle(obj3) :- wheel(obj3, p5)

In this case, FOIL and FOCL show a different behaviour. In fact FOIL converges to the following concept definition:

B: bicycle(A) :- wheel(A, B), wheel(A, C), B<>C

while FOCL does not converge for the same reason as that reported in Section 4.1. The snapshot of the state of FOCL's search on this learning problem is shown in Figure 7.

The different behaviour of FOIL and FOCL is easily explained by the fact that FOIL by default extends the search also to the space of positive and negative literals in the form $X_j = X_k$ (see Section 2.1), while FOCL requires the user/teacher of the system explicitly set the *eql* built-in function in order to do the same, as stated in the previous section. Indeed, when such a parameter is set, FOCL learns the same rule discovered by FOIL:



It is worthwhile to observe that, without introducing the inequality relationship between variables, no concept definition of bicycle consistent with respect to the given training examples exists in the search space. Moreover, the reason for FOCL's non-termination is the same as that given above for the arch problem. These two observations imply that a correct implementation of the conquer stage can provide FOCL with the ability to detect that no correct rule exists and to extend automatically (or interactively) the search to the *difference links* (or *inequality literals*) in the form $X_j \neq X_k$.

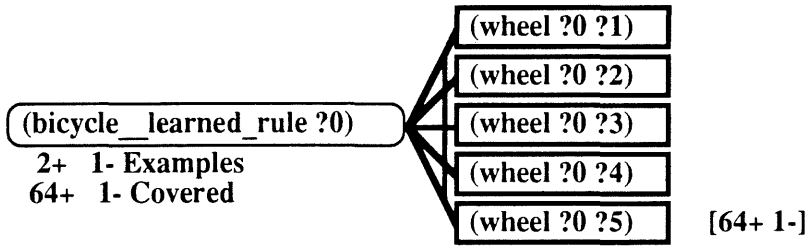


Figure 7. Partial AND/OR graph produced by FOCL for the problem *bicycle* when the training set is TR = {B_1, B_2, N_2 }.

4.3. Learning poker concepts

In a recent paper presenting a new learning system for multiple concept learning, named M-FOCL, Datta and Kibler (1993) describe a problem which consists in learning a subset of the possible five card poker hands. Specifically, they learn the following concepts: *one pair*, *two pair*, *three of a kind*, *four of a kind* and *full house*.

In the paper, the authors consider two alternative representations of the examples.

In the former representation, an example consists of the 5-tuple *(card1, card2, card3, card4, card5)* and they use the predicates *rank(card, r)* and *suit(card, s)* to describe the rank and the suit respectively of each card in the 5-tuple.

In the latter representation, the 1-tuple *(hand)* is used to describe an example. The predicates used in this representation are:

one-pair(H)	H is a hand containing exactly two cards of the same rank and whose suits are different
hand-contains-card(H, C)	C is a card of H
suit(C, S)	S is the suit of C
value(C, V)	V is the rank of C
value-succ(V_1, V_2)	V_2 is the successor of V_1
value<(V_1, V_2)	V_1 is less than V_2

Shortly, the main difference between the two representations is that the former introduces an artificial ordering among the cards in a hand, which causes a combinatorial explosion of the search (Datta and Kibler, 1993, pg. 91), whereas in the latter the relational nature of an example is represented more properly, since an example is one object (hand) which consists of five subparts (cards). The latter representation is undoubtedly a better approximation of reality, as the same authors recognize - *"The latter representation seems more intuitive, since it describes in detail the highly relational characteristics of the domain; ..."* - but the authors claim that both FOCL and M-FOCL have space limitation problems when learning the concepts with such a representation. In particular, they indicate that the reason for the failures lies in the fact that the unbound variable *?card* in each literal of the kind *hand-contains-card(?hand, ?card)* increases the search space by a factor of 5, following a similar explanation given by Quinlan (1990).

Conversely, we claim that the problems FOCL, M-FOCL and FOIL run into when using such a representation have a different origin.

Let us start with the following consideration:

It is easy to observe that the difference between the two alternative representations in the POKER domain is exactly the same as that existing between Quinlan's and our representation of the ARCH problem.

As a consequence, we can hypothesize that the behaviour of both FOIL and FOCL on the *poker* domain will be the same as that shown for the *arch* problem, when the second representation is adopted.

The empirical evaluation of FOIL and FOCL confirms our hypothesis. Indeed, we randomly generated a training set for learning the concept of *one-pair* consisting of four examples. Two of them are hands containing exactly two cards of the same rank and the remaining two examples are hands which do not contain any single pair (this means that they may contain three or four cards of the same rank and with different suits or two cards of the same rank and suit).

FOIL takes .2 seconds to generate no clause, as expected.

FOCL does not converge at all, but an analysis of the ongoing running of FOCL turns out to be again

extremely interesting. Figure 8 is a snapshot of FOCL's search at a time *t*. The partial clause generated shows clearly that FOCL will not converge to any complete and consistent rule. Furthermore, this time, besides performing its search within the same equivalence class of the specialization hierarchy of $(\mathfrak{L}/\sim, \leq)$ rooted into the linked Horn clause *one-pair(H)* $\leftarrow \varnothing$, **FOCL is actually exploring a set of completely meaningless clauses, as that shown in Figure 8, whose English interpretation is:**

"?0 is a pair if there exist 6 hands which contain the same card (?1)."

Even more so, **the *pattern* described by this rule does not occur in any of the training examples given to the learning system (*2nd trap*)**. This happens because FOCL, as well as FOIL, generates and tests all the possible variabilizations of the defined predicates, provided that they do not violate the constraint of linkedness of the current clause, that is, all the variabilizations that contain at least one *old* variable. Among all the possible variabilizations, some of them give rise to *patterns* which do not match at all the structure of the training examples, but nonetheless they are considered promising by the information gain heuristic because the original training examples do not exist anymore as a whole, i.e. as *ground* definite clauses, since they have been *atomized* and stored as tuples of a relational database, according to Quinlan's paradigm (1990) for "Learning Logical Definitions from Relations".

The following example should make clear the limits of an over simplistic use of a relational database to represent the training instances in a system that learns logical theories.

Example 10.

Consider the following two five card poker hands, each one containing an instance of the concept *one-pair* and represented as :

one-pair(H_1) :- hand-contains-card(H_1, J-H), hand-contains-card(H_1, J-S),
hand-contains-card(H_1, 9-H), hand-contains-card(H_1, 8-H),
hand-contains-card(H_1, A-S), ...

one-pair(H_2) :- hand-contains-card(H_2, K-H), hand-contains-card(H_2, K-S),
hand-contains-card(H_2, 10-D), hand-contains-card(H_2, 7-H),
hand-contains-card(H_2, A-S), ...

where, for clarity sake, we neglected to report all the literals whose predicate is different from *hand-contains-card*. Both in FOIL and in FOCL these two examples are represented as the following set of tuples of the relation *hand-contains-card*:

hand-contains-card

<H_1, J-H>
<H_1, J-S>
<H_1, 9-H>
<H_1, 8-H>
<H_1, A-S>

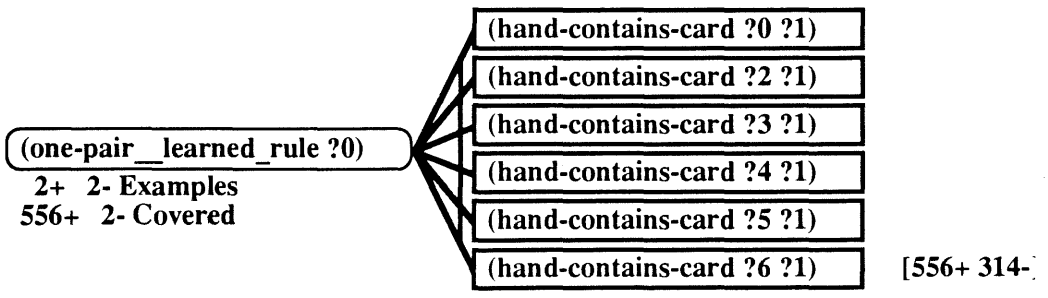


Figure 8. The inconsistent partial clause produced by FOCL for the concept *one-pair* in the *poker* domain. Its English translation is completely counterintuitive.

<H_2, K-H>
<H_2, K-S>
<H_2, 10-D>
<H_2, 7-H>
<H_2, A-S>

When any of these two learning systems performs its search in the hypothesis space, the only *tools* available are the linkedness and the information gain heuristic. Each linked clause defines a *thread* within the relational database that joins all the tuples directly or indirectly related to the arguments in the head of the clause. It is easy to see that if we pull up the thread from the head H_1, all the tuples in the *hand-contains-card* relation and, more precisely, all the tuples in the *poker* database, hang from the thread, because of the ace of spades *(A-S)*.

The example above demonstrates that linkedness alone cannot prevent a first-order learner, that uses a relational database to store the training instances, from losing the structure of an example as a whole and, consequently, from generating and testing all the possible variabilizations of the given primitives, including those that give rise to completely meaningless clauses which do not mirror the structure of any training instance. The knowledge representation consideration underlying the conclusion above also applies to those learning systems from the area of ILP that store the training examples as *facts* in a Prolog database, by splitting the body of an example into its composing literals.

For the sake of completeness, we performed two further experiments with FOIL and FOCL, on the same input data that gave rise to negative results, in order to rule out potential alternative explanations of the reported failures in the *poker* domain.

In the first experiment, we changed the *stopping criterion* of FOIL's conquer stage, which is based on Rissanen's Minimum Description Length (MDL) principle (1983). According to this criterion, an inconsistent clause C is not extended further if the bits required to encode the new clause $D = C \cup \{L_i\}$ exceed the bits needed to indicate the covered tuples[12]. The change we made allows FOIL to find a complete and consistent rule. Indeed, FOIL takes 1450.1 secs to generate the following rule:

```
ONE-PAIR(A)  :- HAND-CONTAINS-CARD(A,B), VALUE(B,C), ~VALUE<(D,C)
ONE-PAIR(A)  :- HAND-CONTAINS-CARD(A,B), SUIT(B,C), VALUE(B,D),
                VALUE-SUCC(E,D), ~VALUE<(F,E), SUIT(F,C),
                HAND-CONTAINS-CARD(G,F), HAND-CONTAINS-CARD(H,F), G<>H,
                ~HAND-CONTAINS-CARD(A,F)
```

Therefore, the cause of the previous negative result seems to be the MDL-based stopping criterion. But a thorough analysis of the results reveals that the rule generated by FOIL is not the correct definition of the concept *one-pair*. Furthermore, the second clause of the rule above describes a pattern that occurs in no positive training examples, if we consider the five card hands as our training examples. The original clause generated by FOIL's conquer stage (before post-pruning) is the following:

```
ONE-PAIR(A)  :- HAND-CONTAINS-CARD(A,B), SUIT(B,C), VALUE(B,D),
                VALUE-SUCC(D,E), VALUE-SUCC(F,D), ~VALUE<(G,F),
                SUIT(G,C), VALUE(G,H), HAND-CONTAINS-CARD(I,G),
                HAND-CONTAINS-CARD(J,G), HAND-CONTAINS-CARD(K,G),
                HAND-CONTAINS-CARD(L,G), I<>J, ~HAND-CONTAINS-CARD(A,G)
```

It reveals that even FOIL's conquer stage performs its search in $(\mathfrak{L}, \leq)$ rather than in the lattice $(\mathfrak{L}/\sim, \leq)$. Indeed, the literals HAND-CONTAINS-CARD(L,G) and HAND-CONTAINS-CARD(K,G) should not be generated at all.

12. This change was made by modifying properly the C language macro `CostOfLit` in the file *defns.i* of FOCL5.0.

As a consequence, we cannot ascribe the previous failure to the stopping criterion of FOIL's conquer stage.

Object identity assumption alone cannot recover this shortcoming. It is needed to store the training instances as ground Horn clauses in a data structure that preserves the intrinsically relational nature of the examples. Then, the positive and negative examples covered by a clause should be computed with respect to this structure, rather than with respect to the tuples in the relational database.

In the second experiment, we exploited the possibility of FOCL of using constraints to limit the search space. Specifically, we reduced the number of variabilizations of a predicate, that are generated and tested during the conquer stage, by declaring properly the *mode* of the predicate. Mode information is associated with each argument of a predicate and indicates whether that argument must be an old variable (an *input* variable, according to logic programming terminology) or may be a new one[13]. From the previous experiments, it is clear that the problems of FOIL and FOCL with the *poker* domain come from the fact that these systems are not able to rule out unrealistic variabilizations of the predicate *hand-contains-card*, such as:

```
ONE-PAIR(A)    :- ..., HAND-CONTAINS-CARD(G,F), HAND-CONTAINS-CARD(H,F),
                  G<>H, ...
```

Therefore, we used the mode in order to specify that the first argument of the predicate *hand-contains-card* must be an old variable. Figure 9 shows the rule produced by FOCL with such a parameter setting. Some literals in the rule have been deleted by the phase of clause simplification, which post-processes each clause in a rule to eliminate any literal that turns out to be unnecessary. This phase is needed to overcome the *greediness* of the hill-climbing search strategy, which sometimes finds local maxima rather than global ones. It is worthwhile to note that the second clause of the rule is the correct concept definition of *one-pair*, that is, it is consistent and complete w.r.t. the given training examples. Therefore, the first clause in the rule turns out to be redundant. This suggests straightway the need of having in FOIL and FOCL also a smart post-processor of rules that inspects each clause within a rule and deletes redundant clauses.

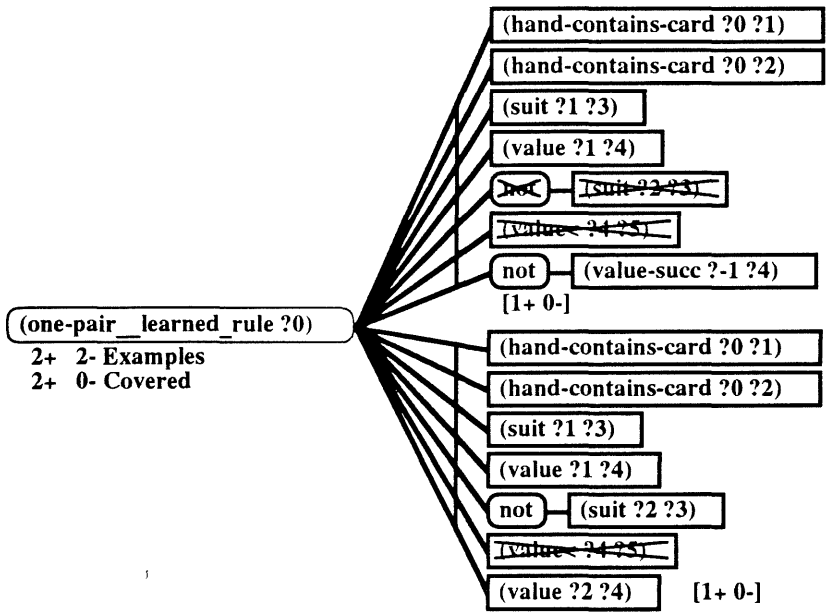


Figure 9. The rule produced by FOCL for the concept *one-pair* in the *poker* domain by forcing the first argument of *hand-contains-card* to be an old variable.

13. There are three mode specifications: :+, :- and :? indicating that the argument must, must not and could or not be bound, respectively.

4.4. Document understanding

Esposito, Malerba, Semeraro, and Pazzani (1993a) describe a real world problem that can be addressed by means of inductive learning techniques. The problem is document understanding and consists in learning classification rules for the identification of logical components in the page layout of any office document. Indeed, according to the ODA/ODIF standard (Horak, 1985), any document is characterized by two different structures representing both its content and its internal organization: The *layout* (or *geometric*) structure and the *logical* structure.

The former associates the content of the document with a hierarchy of *layout objects* such as text lines, vertical/horizontal lines, graphic elements, photographic elements, columns, pages and so on (Figure 10). The latter associates the content of the document with a hierarchy of *logical objects* such as title, abstract, paragraphs, sections, chapters, tables, figures, footnotes, page number and so on (Figure 11).

The layout/logical structure of a document is described by a set of attributes, that give the characteristics of each object, as well as relations among different objects. For instance, layout objects can be characterized by the type of content (text, graphics, etc.), their position in the page, their shape, their dimension, as well as the numerical properties of their bitmaps, while logical objects can be described by their type (abstract, paragraph, etc.), some key-words contained in the text (date, figure,

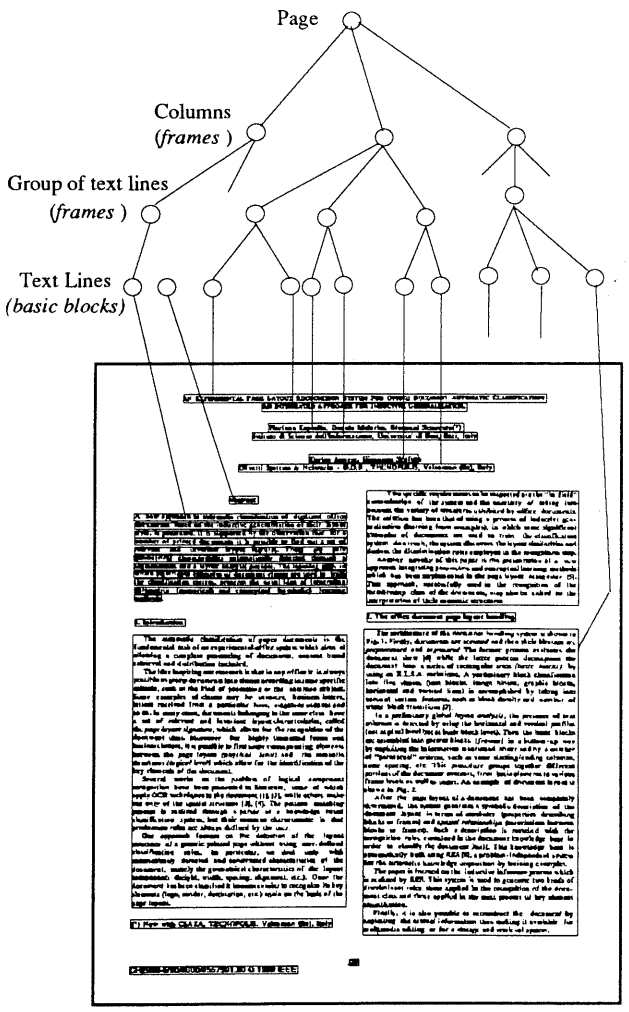


Figure 10. The hierarchical layout structure of a document.

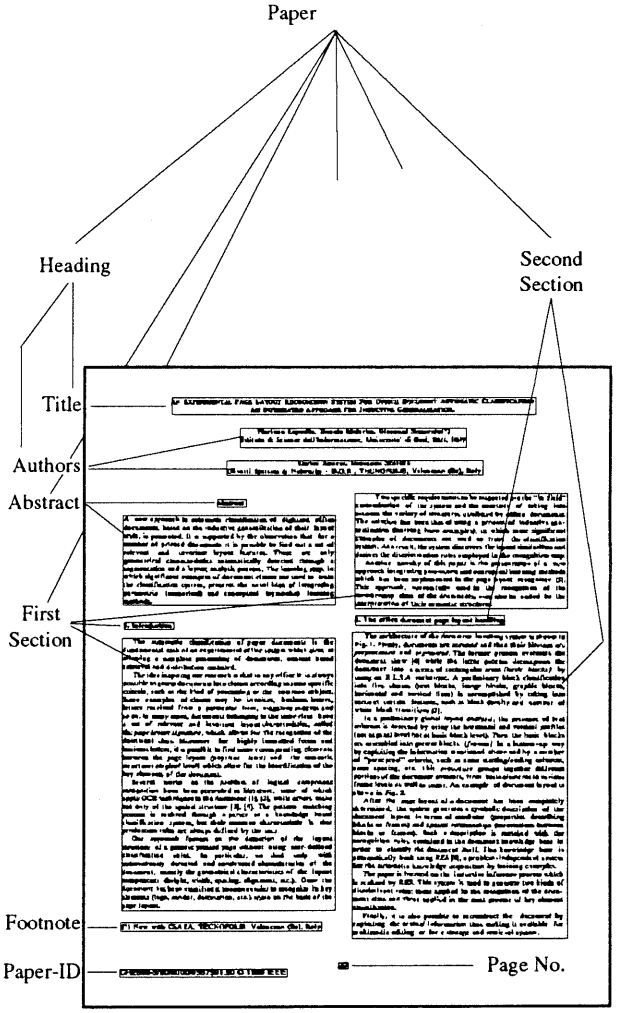


Figure 11. The hierarchical logical structure of a document.

etc.), and their position. Several kinds of relationships among objects can be defined. Of course, the hierarchy in the layout/logical structures defines some hierarchical relationships among objects of the same structure. However, other, and perhaps more interesting, relationships exist among logical objects (*logical-logical* relationships) and among layout objects (*layout-layout* relationships). An example of layout-layout relationship is the mutual position of two layout objects, while the cross-reference of a caption to a figure or the reading order of some parts of a document are two examples of logical-logical relationships. Finally, *logical-layout* relationships between one or more elements of the layout hierarchy and one element of the logical hierarchy can be defined. These last are the most interesting, since they allow us to identify some logical components of a document without reading its content by means of an optical character recognizer (OCR) but using only layout (or geometrical) characteristics. For instance, in a standard English letter, the date is under the sender's address which is in turn in the top left hand corner. This simple layout information can be profitably exploited by a document management system to identify specific portions of content (INTREPID, 1992). Therefore, when there exist logical-layout relationships, due to a standard format of the document, then it is possible to *understand* a document by using only layout information extracted from the layout analysis process.

The complete set of predicates used to describe the layout/logical structure of the documents is reported in Table II.

The problem of document understanding can be cast as a learning problem. Given a set of documents whose page layouts have already been analyzed and assumed that some layout objects have been correctly labelled according to their meaning - for instance, as a sender or receiver block in a letter - then the classification rules for the identification of logical objects in other (previously unseen) documents can be learned inductively.

The main advantage of adopting a machine learning approach for this problem is a greater flexibility of the office document management system since it can be customized more quickly and easily. Nevertheless, document understanding is a complex learning problem. The main source of complexity lies in the necessity of exploiting *contextual information*

Table II

Predicates for the Page Layout Description

Predicate	Meaning
logic_type-sender(X) logic_type-receiver(X) logic_type-logo(X) logic_type-ref(X) logic_type-date(X) logic_type-unsigned(X)	logical label of the layout object X
width-very-very-small(X) width-very-small(X) width-small(X) width-medium-small(X) width-medium(X) width-medium-large(X) width-large(X) width-very-large(X) width-very-very-large(X)	width of the layout object X
height-smallest(X) height-very-very-small(X) height-very-small(X) height-small(X) height-medium-small(X) height-medium(X) height-medium-large(X) height-large(X) height-very-large(X) height-very-very-large(X) height-largest(X)	height of the layout object X
type-text(X) type-hor-line(X) type-picture(X) type-ver-line(X) type-graphic(X) type-mixture(X)	type of the layout object X
part_of(X, Y)	layout object Y belongs to document X
position-top-left(X) position-top(X) position-top-right(X) position-left(X) position-center(X) position-right(X) position-bottom-left(X) position-bottom(X) position-bottom-right(X)	position of the layout object X
on_top(X, Y)	layout object X is on top of layout object Y
to_right(X, Y)	layout object X is to the right of layout object Y
aligned-only-left-col(X, Y) aligned-only-right-col(X, Y) aligned-only-middle-col(X, Y) aligned-both-columns(X, Y) aligned-only-upper-row(X, Y) aligned-only-lower-row(X, Y) aligned-only-middle-row(X,Y) aligned-both-rows(X, Y)	layout objects X and Y are aligned

during the learning process. Indeed, concepts to be learned refer to parts of a document rather than to the entire document and, even more so, they are not independent of each other since parts of a document may be related to each other according to logical-logical relationships. Therefore document understanding is a *multiple-concept* learning problem which cannot be cast as a finite number of canonical *single-concept* learning problems, as most of the existing multi-concept learners do on the ground of the implicit assumption that concepts to be learned are independent (*independence assumption*). More precisely, document understanding is an instance of the more general problem of *learning mutually dependent concepts* or *Contextual Supervised Learning* (CSL), that requires a learning paradigm different from the canonical one (Michalski, 1983).

Of course, traditional learning algorithms making the independence assumption can still be exploited for the problem of document understanding by simply neglecting logical-logical relationships, but, as reported in (Esposito *et al.*, 1993a), the empirical comparison of the traditional learning strategies with new strategies for CSL shows that, by dropping the independence assumption and taking into account concept dependencies, it is possible to generate more accurate and simpler rules, since the learning paradigm is a better approximation of reality, in spite of a kind of *ripple effect* affecting the classification process, for which any misclassification due to an overgeneralized/overspecialized rule is propagated to all the dependent rules.

In the domain of document understanding, we considered a set of 30 single page documents, namely copies of letters sent by Olivetti.

The task consists in learning 5 concepts, namely *sender* of the letter, *receiver*, *logotype*, *reference number*, and *date*.

Six different experiments were performed. For each experiment, the set of 30 documents was randomly split into two subsets in six different ways: A training set of 20 documents and a testing set of 10 documents. Figure 12 shows an example of an Olivetti letter, in which the layout objects have been already detected by the phase of *layout analysis* and labelled by the teacher/user of the learning system according to their meaning. Each object (block) in the layout structure of a document constitutes a training/testing example. Figure 13 reports the complete description of the sender block for the letter in Figure 12.

(Esposito *et al.*, 1993a) reports the complete set of 36 experiments performed by exploiting FOCL's capability of using background knowledge (*intensionally defined* predicates) and/or relational clichés, and of producing recursive rules.

In two cases, FOCL was not able to produce any complete and consistent rule for the concept *date*. More precisely, in the last of the six experiments performed by using both the available background knowledge and four clichés, FOCL was not able to generate a rule for the concept date after almost 4 hours of CPU time on a SUN station 4/25. Indeed, FOCL preferred to introduce predicates with new variables (more than 10), enormously widening the search space at each step without really improving consistency. The existence of a simpler and consistent rule is guaranteed by the fact that in the previous experiments with only 2 clichés FOCL always converged towards a solution. FOCL showed a similar behaviour in the last of the six experiments made by setting FOCL's parameters to allow recursive rules and using again the background knowledge and the four clichés. Table III reports the number of training and testing examples used in both the experiments that produced negative results.

The reason of FOCL's divergence is the same as for the arch and the bicycle problems: It searches in the wrong specialization hierarchy.

This results points out that the conceptual deficiency of FOCL's conquer stage may become manifested in real world learning tasks, in addition to laboratory-sized ones.

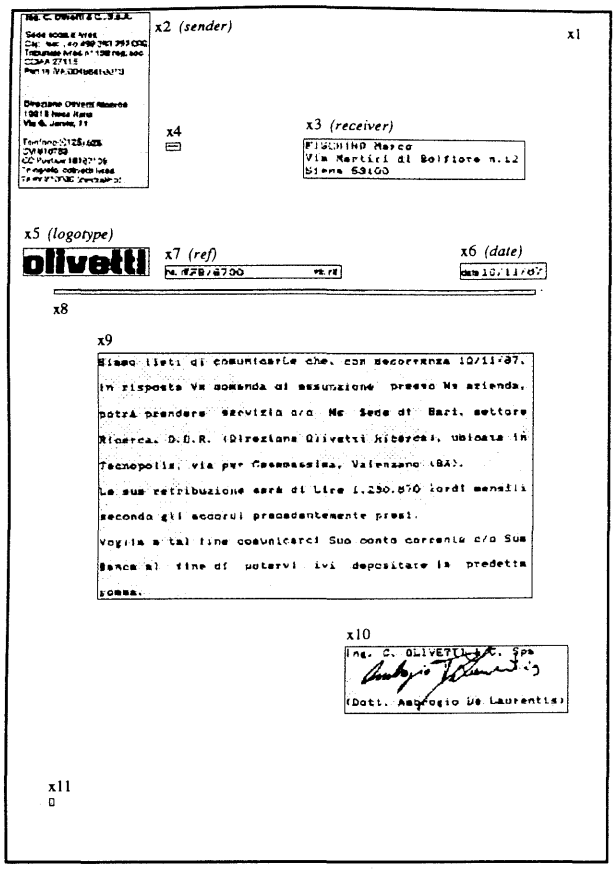


Figure 12. Page layout of an Olivetti letter with labelled blocks.

logic_type-sender(x2) ←
logic_type-receiver(x3), logic_type-unsigned(x4),
logic_type-logo(x5), logic_type-date(x6), logic_type-ref(x7),
logic_type-unsigned(x8), logic_type-unsigned(x9),
logic_type-unsigned(x10), logic_type-unsigned(x11),
part_of(x1,x2), part_of(x1,x3), part_of(x1,x4), part_of(x1,x5),
part_of(x1,x6), part_of(x1,x7), part_of(x1,x8), part_of(x1,x9),
part_of(x1,x10), part_of(x1,x11),
width-medium(x2), width-medium-large(x3), width-smallest(x4),
width-medium(x5), width-medium-small(x6),
width-medium-large(x7), width-very-very-large(x8),
width-very-very-large(x9), width-medium-large(x10),
width-smallest(x11),
height-medium-large(x2), height-small(x3), height-smallest(x4),
height-very-small(x5), height-very-very-small(x6),
height-very-very-small(x7), height-smallest(x8), height-large(x9),
height-medium-small(x10), height-smallest(x11),
type-text(x2), type-text(x3), type-text(x4), type-picture(x5),
type-text(x6), type-text(x7), type-text(x8), type-text(x9),
type-mixture(x10), type-text(x11),
position-top-left(x2), position-top(x3), position-top-left(x4),
position-top-left(x5), position-top-right(x6), position-top(x7),
position-center(x8), position-center(x9), position-bottom-right(x10),
position-bottom-left(x11),
on_top(x5,x8), on_top(x6,x8), on_top(x7,x8), on_top(x9,x10),
to_right(x2,x4), to_right(x5,x7),
aligned-both-columns(x2,x5), aligned-only-lower-row(x5,x7),
aligned-only-left-col(x4,x7), aligned-both-rows(x7,x6),
aligned-only-right-col(x8,x9), aligned-only-upper-row(x4,x3),
aligned-only-left-col(x8,x11)

Figure 13. Ground Horn clause for the sender of the layout in Figure 12.

Table III

Training and testing examples for each class used in two experiments with FOCL

	Training set	Test set
logo	20	10
sender	25	10
ref	33	12
date	24	13
receiver	28	11
unsigned	124	54
TOTAL	254	110

5. Theoretical solution and object identity

In this section, we consider the negative results of FOIL and FOCL and propose theoretical solutions that prevent these systems from going through the troubles shown in Section 4. Moreover a practical solution is proposed and implemented in a new version of FOCL, namely FOCL-1-2-3 ver.1.10. The use of this new version of FOCL on the same tasks reported in Section 4 shows the strengths and the weaknesses of the practical solution in comparison with the theoretically founded ones.

The theoretical solution to the first kind of trap is to change the search space of the conquer stage of FOIL and FOCL from $(\mathfrak{L}, \leq)$ to $(\mathfrak{L}/\sim, \leq)$.

Practically speaking, this means to provide the conquer stage of the high level separate-and-conquer strategy with a procedure that performs a θ-subsumption equivalence test. This procedure should check if the current clause is equivalent (under $\sim$) to the clause obtained by adding the literal (or a combination of literals, if clichés are used) with the highest information gain. In the general case, this is an NP-complete problem, since it involves a test for θ-subsumption, which is NP-complete (Garey and Johnson,

1979). In the particular case of FOIL and FOCL, the current clause C is a proper subset of the clause D, obtained by adding the literal with the maximum information gain, since the conquer stage performs a general-to-specific search. Thus, the procedure needs to check only if D θ-subsumes C.

In formulae, $C \subset D$ and we want to know if $C \sim D$. But $C \subset D$ implies that $D \leq C$, thus it is enough to check if $C \leq D$. By Definition 4, $C \leq D \Rightarrow \exists\, \sigma : D\sigma \subseteq C$.

$D\sigma \subseteq C$ means that for each literal D_i in D there is a corresponding literal C_j in C such that $D_i\sigma \equiv C_j$. Therefore, in the worst case, it is necessary to perform $(|D| - 1) * (|C| - 1) + 1$ unifications. Since $|D| = |C| + 1$ (with the exception of the case in which a cliché has the maximum information gain), $|C|^2 - |C| + 1$ unifications of function-free literals are required. From the literature about unification (Siekmann, 1990; Knight, 1989), it is known that the best unification algorithms have a linear complexity in time (Paterson and Wegman, 1978; Martelli and Montanari, 1982). In conclusion, in the worst case, the procedure that performs the θ-subsumption equivalence test has a complexity in time equal to $O(n^2)$, where n is the *size* of the current clause.

The computational complexity of such a procedure can be reduced to $O(n)$ by observing that, for the *conquer* stage of FOIL and FOCL, the following proposition holds:

Proposition 3.
Let C_1 and C_2 be two definite Horn clauses:

$$C_1 \text{ reduced}, C_1 \subset C_2 \text{ and } vars(C_1) = vars(C_2) \Rightarrow [C_1]_{\sim} \neq [C_2]_{\sim}$$

Proof:
Ad absurdum.
Let us suppose that C_1 and C_2 are two definite Horn clauses s.t.:
C_1 is reduced, $C_1 \subset C_2$, $vars(C_1) = vars(C_2)$ (i.e., C_2 does not introduce new variables) and $[C_1]_{\sim} = [C_2]_{\sim}$.
Without loss of generality we can suppose $C_1 = \{L_1, L_2, \ldots, L_n\}$ and $C_2 = \{L_1, L_2, \ldots, L_n, L_{n+1}\}$, so that $C_2 = C_1 \cup \{L_{n+1}\}$, where L_{n+1} is necessarily a literal in the body of C_2. Furthermore, $vars(C_1) = vars(C_2) \Rightarrow vars(\{L_{n+1}\}) \subseteq vars(C_1)$ (3)
By definition, $[C_1]_{\sim} = [C_2]_{\sim}$ means $C_1 \leq C_2$ and $C_2 \leq C_1$.
$C_1 \leq C_2 \Rightarrow \exists\, \sigma : C_2\sigma \subseteq C_1 \Rightarrow H_2\sigma \equiv H_1$, where as usual H_i denotes the head of the clause C_i.
$C_2 \leq C_1 \Rightarrow \exists\, \mu : C_1\mu \subseteq C_2 \Rightarrow C_1\mu \subseteq C_1 \cup \{L_{n+1}\} \Rightarrow H_1\mu \equiv H_2$. It suffices to consider $\mu = \{\}$.
Therefore $H_1 \equiv H_2$. But $H_2\sigma \equiv H_1$ holds, thus σ does not map any variable in H_2.
$C_2\sigma \subseteq C_1 \Rightarrow (C_1 \cup \{L_{n+1}\})\sigma \subseteq C_1 \Rightarrow C_1\sigma \subseteq C_1$ and $\{L_{n+1}\}\sigma \subseteq C_1$.
Let us consider $C_1\sigma$.
We know that $C_1\sigma \subseteq C_1$ (4)
$C_1\sigma \subseteq C_1 \Rightarrow C_1 \leq C_1\sigma$ and, obviously, it holds also $C_1\sigma \leq C_1$ (since $C_1\sigma = C_1\sigma \Rightarrow C_1\sigma \subseteq C_1\,\sigma$), thus $C_1 \sim C_1\sigma$ (5)
Now, either $C_1\sigma \subset C_1$ or $C_1\sigma = C_1$ holds.
If $C_1\sigma \subset C_1$ holds, then $C_1\sigma \neq C_1$. But (4) and (5) and $C_1\sigma \neq C_1$ imply that C_1 is not reduced, which contradicts the first hypothesis.
If $C_1\sigma = C_1$ holds, then it suffices to consider $\sigma = \{\}$. But, then $\{L_{n+1}\}\sigma \subseteq C_1$ is equivalent to say that $\{L_{n+1}\} \subseteq C_1$ by (3), which contradicts the hypothesis that $C_1 \subset C_2$.
As a consequence, it holds $[C_1]_{\sim} \neq [C_2]_{\sim}$ Q.E.D.

In other words, it is necessary to perform a θ-subsumption equivalence test iff the new clause D introduces *new* variables. Therefore, since $D = C \cup \{L_{n+1}\}$, where L_{n+1} is the literal with maximum information gain, it is sufficient to search for a unification between L_{n+1} and one of the n literals in C, and this search has a linear complexity in the size of the clause. The trivial case in which $L_{n+1} \in C$ can be easily ruled out by preventing the search from adding a literal that already exists in the body of the currently generated clause.

The second trap, detected in the *poker* domain, is not solved by the change of the search space from $(\mathfrak{L}, \leq)$ to $(\mathfrak{L}/\sim, \leq)$. In other words, the θ-subsumption equivalence test is not enough to avoid this kind of problem. It requires further processing.

Some existing learning systems avoid such a trap by adopting different shrewdnesses.

For instance, ML-SMART (Bergadano and Giordana, 1988) uses a relational database to store the training instances, but each tuple in the database is actually a pointer to another structure which contains the original examples, represented as ground Horn clauses.

The systems by Winston (1970), Hayes-Roth (1974) and Hayes-Roth and McDermott (1977), and Vere (1975; 1977; 1980) represent the examples as graphs or equivalent formalisms (semantic networks, parameterized structural representations, conjunctions of literals, respectively). Moreover, they are incremental (or *data-driven*) systems that learn lgg's (or *Maximally-Specific Conjunctive generalizations*).

The family of INDUCE learning systems (Larson, 1977; Dietterich, 1983; Bentrup *et al.*, 1987) and INDUBI (Semeraro, 1988; Esposito, 1990) represent the examples as decision rules, a representation language equivalent to *colored* graphs (Stepp, 1984). Furthermore, they adopt the generalization model defined by the operation of subgraph isomorphism (equivalent to a restriction of θ_{OI}-subsumption to decision rules) to prevent the first kind of traps and the concept of *seed* to prevent the second kind of traps. A new system, called INDUBI/H (Esposito *et al.*, 1993d), represents the examples as ground definite clauses and fully adopts the model of generalization defined by θ-subsumption under object identity. INDUBI/H is an extension of INDUBI in several aspects. The main extension allows INDUBI/H to learn linked (definite) Horn clauses instead of decision rules, that can be viewed as Horn clauses with no head, i.e. variables are not allowed to occur in the action part of the rule.

As to FOIL and FOCL, the solution adopted by ML-SMART seems the simplest to be implemented among all those listed above, since all these three systems adopt the same internal representation of the examples.

In order to overcome the first kind of problems, we changed the underlying model of generalization of FOCL from θ-subsumption to θ_{OI}-subsumption. As a consequence, the search space of FOCL changes from $(\mathfrak{L}, \leq)$ to $(\mathfrak{L}, \leq_{OI})$. It is worthwhile to observe that such a solution is not equivalent to the theoretical one, but in some cases it eliminates the problem of non-termination, since it allows the search to move from an equivalence class to a distinct one whenever the literal with maximum information gain introduces a *new* variable. Nevertheless, if this literal does not introduce any new variable, Proposition 3 guarantees that the search is not performed within the same equivalence class. In other words, this solution does not cover all the range of problems in which FOIL and FOCL can go through, but it represents an *approximate* solution that is very easy to implement in FOCL.

This solution has been implemented in a new version of the system, namely FOCL-1-2-3 ver.1.10. This new version of FOCL allows the end-user/teacher to adopt the θ_{OI}-subsumption generalization model in a modular way. Indeed, the user can choose the type(s) of variables for which object identity is assumed.

5.1. Learning the concept of an arch under object identity

Figure 14 shows the complete AND/OR graph produced by FOCL-1-2-3 ver. 1.10 with the same input data - training set and system parameters - as those used in the experiment described in Section 4.1., but changing properly the typing information in order to assume object identity for the variables that represent a *part* of an arch. The shaded literals in the AND/OR graph are the inequality literals added by FOCL-1-2-3 ver.1.10 to a clause whenever a literal introduces a new variable whose type is *part*.

Object identity assumption allows FOCL to prevent the problem of non termination of the search

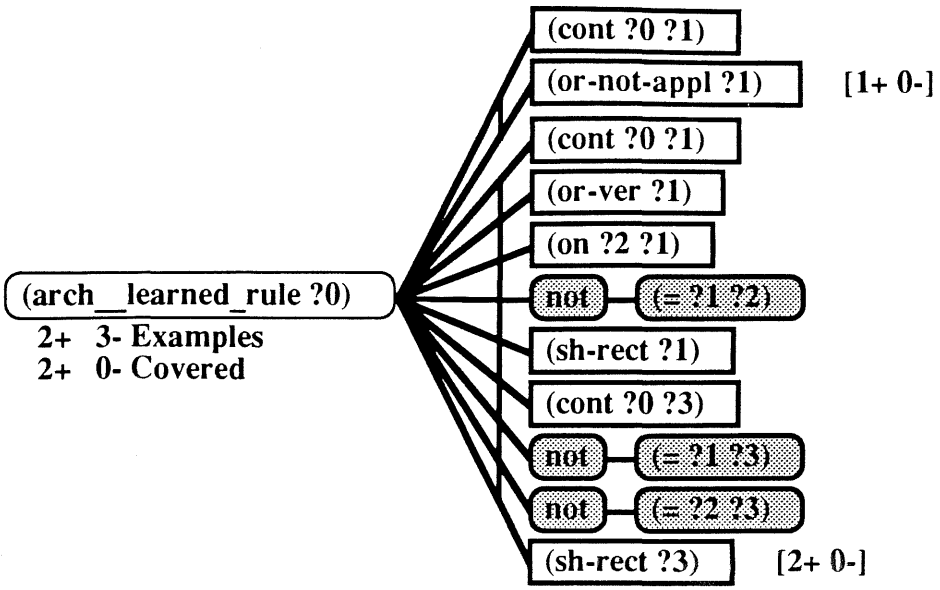


Figure 14. Complete and consistent definition for the concept *arch* produced by FOCL-1-2-3 ver. 1.10 by changing the underlying model of generalization from θ-subsumption to θ_{OI}-subsumption. The shaded literals are the *inequality literals* which are explicitly added by FOCL-1-2-3 ver. 1.10 to the Horn clause representation of a concept definition.

when learning a complete and consistent rule for the concept of *arch*.

5.2. Learning the concept of a bicycle under object identity

Figure 15 shows the rule generated by FOCL-1-2-3 ver. 1.10 when θ_{OI}-subsumption is assumed as generalization model for the variables whose type is *wheel*. It is worthwhile to note that the concept definition produced is the same as that found by FOCL when the *eql* built-in function is set.

Again, object identity assumption allows FOCL to recover the relevant property concerning the number of wheels, with no user intervention (to set the *eql* built-in function), and to overcome the problem of divergence of the conquer stage.

5.3. Learning poker concepts under object identity

In the poker domain we changed the generalization model from θ-subsumption to θ_{OI}-subsumption both for the variables whose type is *hand* and for the variables whose type is *card*.

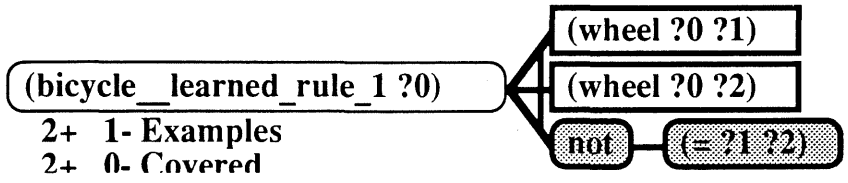


Figure 15. Complete and consistent definition for the concept *bicycle* produced by FOCL-1-2-3 ver. 1.10 by changing the generalization model for the variables whose type is *wheel* from θ-subsumption to θ_{OI}-subsumption.

As expected, this time object identity assumption alone cannot recover the negative results obtained before on this learning task. More precisely, θ_{OI}-subsumption model of generalization allows FOCL-1-2-3 ver.1.10 to overcome the problem of divergence of the conquer stage, as guaranteed by the theoretical analysis made above, but it cannot prevent the system from producing meaningless, even though complete and consistent, concept definitions, like that one shown by Figure 16. Nonetheless, object identity assumption does not jeopardize the good results obtained by FOCL using the *mode* constraint. Indeed, Figure 17 shows the rule generated by FOCL-1-2-3 ver.1.10 with the same parameter setting as that used to generate the rule in Figure 9, but making the object identity assumption both for the variables whose type is *hand* and for the variables whose type is *card*. The comparison of the two rules points out that there are no significant differences. The only difference concerns the deletion of the literal *not (suit ?2 ?3)*, which becomes useless, on the ground of the given training examples, when θ_{OI}-subsumption is assumed as generalization model, since the variables *?1* and *?2* can be no longer bound to the same card in a hand.

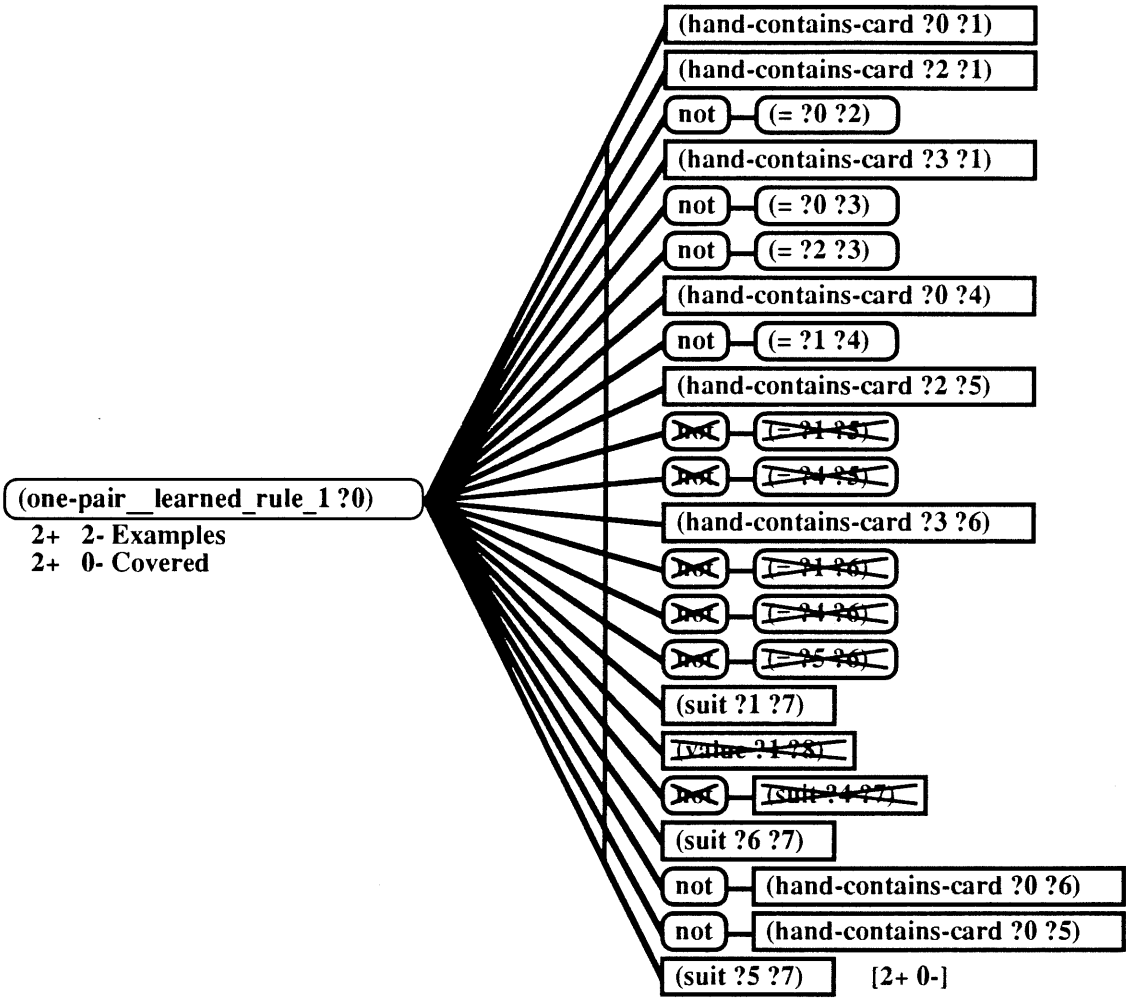


Figure 16. Complete and consistent definition for the concept *one-pair* produced by FOCL-1-2-3 ver. 1.10 when θ_{OI}-subsumption (rather than θ-subsumption) is assumed as generalization model both for the variables whose type is *hand* and for the variables whose type is *card*. Some literals have been deleted by the process of clause simplification.

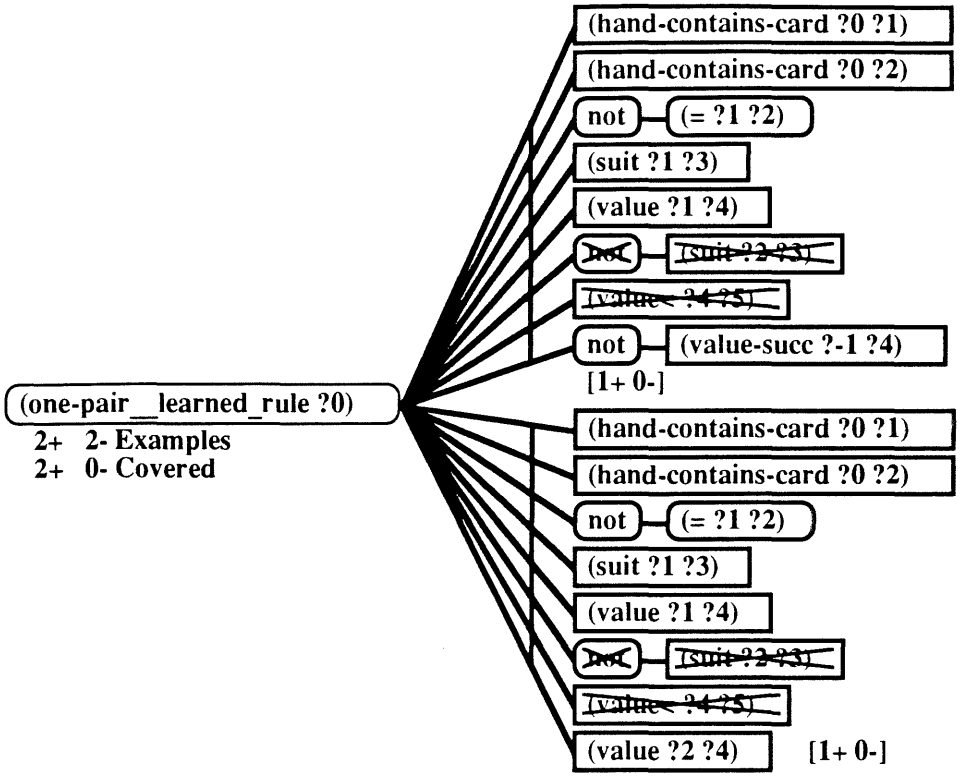


Figure 17. Rule produced by FOCL-1-2-3 ver. 1.10 for the concept *one-pair* when θ_{OI}-subsumption is assumed as generalization model both for the variables whose type is *hand* and for the variables whose type is *card* and the mode of the first argument of the predicate *hand-contains-card* is set to :+.

5.4. Document understanding under object identity

As to the learning problem of document understanding, we repeated the 36 experiments reported in (Esposito *et al.*, 1993a) by adopting θ_{OI}-subsumption as generalization model.

The results show that rules produced by FOCL-1-2-3 ver.1.10 for all the 34 cases in which FOCL-1-2-3 ver.1.1 converged are exactly the same as the rules produced by dropping object identity assumption. Furthermore, object identity allows FOCL to converge also in those two cases in which it was not able to generate any rule for the concept *date*. In particular, FOCL-1-2-3 ver.1.10 generated the same rule both in the last of the six experiments made by using the background knowledge and four clichés and in the last of the six experiments performed by setting FOCL's parameters to generate recursive rules and using again the background knowledge and the same four clichés as before. Figure 18 shows this rule.

6. Conclusions and future work

All learning systems need to clarify the adopted model of generalization and the space in which they perform the search for concept definitions. Formal methods and techniques, like those used in the area of logic programming, can be useful to detect potential sources of problems and conceptual shortcomings of the existing empirical (and integrated) learning systems and to suggest the suitable counteractions in order to improve their performance.

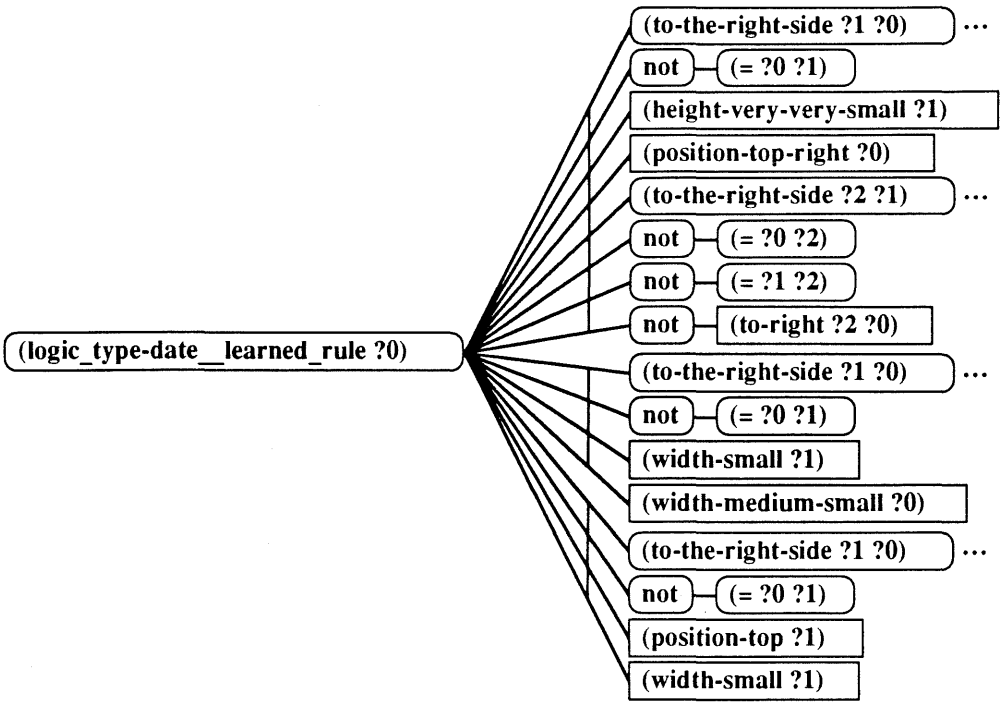


Figure 18. Rule produced by FOCL-1-2-3 ver. 1.10 for the concept *date* under object identity assumption in the two experiments in which the system did not converge by dropping such assumption. The symbol ... denotes the literals in which an intensionally defined predicate occurs.

This paper constitutes an attempt to apply this analytical approach to two well-known learning systems, FOIL and FOCL. Plotkin's logical framework to inductive generalization (1970), properly revised to be adapted to the characteristics of the logical language used by these learning systems (and many others), proved useful both to point out some lacks that affect the search strategy of these systems and to suggest straightforwardly the adequate corrections. These corrections have been partially implemented in a new version of FOCL, namely FOCL-1-2-3 ver. 1.10, and proved effective to overcome some of the detected problems. Future work will concern the full implementation of the theoretically-founded mechanisms to overcome the detected problems and the empirical test of these mechanisms both on those learning tasks purposely used to stress FOIL and FOCL and on real world domains. Moreover, we intend to implement a new version of FOCL in which, at first, the θ_{OI}-subsumption generalization model is fully adopted (for any type of variables) and then the system autonomously learns the *equality literals* in the form $[X_i=X_j]$, if necessary. Indeed, object identity assumption turns out to be theoretically limitative when the concepts to learn are described by reflexive predicates and, more generally, when this kind of predicates are used (as intensionally or extensionally defined predicates). For instance, when learning the concept of *murder(x, y)*, we should be able to cover also instances of suicides, that is, ground Horn clauses whose head is *murder(g, g)*, where *g* is a constant of the language, or alternatively to recognize these instances as particular cases of murdering and generate a proper rule for the subclass *murder(x, x)* according to an *object-oriented* concept-learning scheme. Conversely, by assuming object identity, the learning system is compelled to generate a distinct clause for the training instances of the kind *murder(g, g)*. Even though this behaviour is considered by some authors undesirable, it may lead to discover new concepts as subclasses of the concepts to learn. In the previous example of *murdering*, it leads to discover the concept *suicide(x)*.

Theoretically, this new learning strategy should prove more efficient than that currently implemented

by FOCL-1-2-3 ver. 1.10, since θ_{OI}-subsumption is more manageable than θ-subsumption, while learning the equality literals would happen rarely, according to a *when-needed* strategy. Again, the empirical comparison of the results produced by this new system to the results produced by FOCL-1-2-3 ver. 1.10 will provide us with new hints about the effectiveness of this learning strategy.

Furthermore, this analytic approach to concept learning allows us to identify the limits of commonly accepted theories and methodologies for inductive learning and to critically revise and reformulate them on the ground of the cited logical frameworks. For instance, we can now state that, under object identity, adding a literal to a clause is always a specializing rule (unless the added literal already exists in the clause), while, by dropping such assumption, this is no longer true. More precisely, it is true iff the new clause is not equivalent to the previous one. In a similar manner, the *dropping condition* generalization rule (Michalski, 1983), holds under θ_{OI}-subsumption, but it is no longer a generalization rule under θ-subsumption if the new clause belongs to the same equivalence class as the previous one. The immediate consequence of this observation is that the generalization rules in (Michalski, 83) should be reformulated (indeed, Michalski uses implication, i.e. subsumption, as generalization model and the same observation applies also to implication) or, alternatively, it should be stated clearly that they hold under θ_{OI}-subsumption.

Another useful *side effect* of this analytic approach to concept learning is the development of some optimization techniques that can improve the efficiency of the learning algorithms by pre-processing the training instances or post-processing the produced rules. For instance, under the generalization model defined by θ_{OI}-subsumption, the *turning constants into variables* generalization rule (Michalski, 1983) becomes deterministic and can be used to prune the training set of those examples that are useless for the learning process, according to the following observation:

Let E_1 and E_2 be two positive examples of the concept P and G any rule for P. By definition, $E_i \leq_{OI} G$ and $E_i\gamma \leq_{OI} G$, i=1,2, where γ is an *inductive substitution* (Vere, 1980) (or *antisubstitution* (Knight, 1989)) that turns each distinct constant in E_i into a distinct variable, according to the *turning constants into variables* rule. Then, if it holds that $E_1\gamma \leq_{OI} E_2\gamma$, then $E_1\gamma \leq_{OI} E_2\gamma \leq_{OI} G$, and E_1 is useless for learning G.

This pre-processing technique has been successfully implemented in INDUBI/H (Esposito *et al.*, 1993d), together with the related post-processing technique that allows INDUBI/H to simplify the learned rules by deleting those clauses that are *logically redundant,* since they are θ-subsumed by other clauses in the same rule. This technique could be exploited by any learning system that adopts a high level *separate-and-conquer* strategy, such as AQ, INDUCE, FOIL and FOCL.

Acknowledgments

This work has been carried out when Giovanni Semeraro was visiting the machine learning group of Prof. Michael Pazzani at the University of California, Irvine.

The authors would like to thank Lorenza Saitta, Kamal Ali, and Luc De Raedt for constructive criticism on this work, Cesare Tinelli for the help in proving Proposition 1, Piew Datta for commenting on an earlier draft of this paper, and Peter Idestam-Almquist for suggesting Example 2 and for helpful discussions on the formal topics of this work. Thanks also to all the graduate students of the Artificial Intelligence Research Center at University of California, Irvine for helpful comments on the ideas presented in this paper. Document understanding data set was provided by Enrico Annese from Olivetti Systems & Networks.

A special thank to Prof. Floriana Esposito for supporting Giovanni Semeraro during his stage in Irvine and for continuously stimulating his research and to Donato Malerba for several discussions on the topics of this paper.

References

(Apt, 1990) Apt, K.R., Introduction to Logic Programming, in *Handbook of Theoretical Computer Science*, J. van Leeuwen (Ed.), 492 - 574, Elsevier Science Publisher, 1990.

(Bain and Muggleton, 1991) Bain, M., and Muggleton, S., Non-Monotonic Learning, in *Machine Intelligence 12*, J.E. Hayes-Michies and E. Tyugu (Eds.), Oxford University Press, 1991 (also in *Inductive Logic Programming*, S. Muggleton (Ed.), 145-161, Academic Press, San Diego, CA, 1992).

(Barbuti *et al.*, 1990) Barbuti, R., Mancarella, P., Pedreschi, D., and Turini, F., A Transformational Approach to Negation in Logic Programming, *Journal of Logic Programming*, 8, 201-228, 1990.

(Bell and Weber, 1993) Bell, S., and Weber, S., On the close logical relationship between FOIL and the frameworks of Helft and Plotkin, *Proceedings of The Third International Workshop on Inductive Logic Programming ILP'93*, Bled, Slovenia, 1-10, 1993.

(Bergadano *et al.*, 1988) Bergadano, F., Giordana, A., and Saitta, L., Automated concept acquisition in noisy environments, *IEEE Transactions on Pattern Analysis and Machine Intelligence*, Vol. PAMI-10, 555-578, 1988.

(Bergadano and Giordana, 1988) Bergadano, F., and Giordana, A., A Knowledge Intensive Approach to Concept Induction, *Proceedings of the Fifth International Conference on Machine Learning*, Ann Arbor, Michigan, 305-317, 1988.

(Bentrup *et al.*, 1987) Bentrup, J.A., Mehler, G.J., Riedsel, J.D., *INDUCE 4: A Program for Incrementally Learning Structural Descriptions from Examples*, Department of Computer Science, University of Illinois at Urbana-Champaign, Urbana, Illinois, UIUCDCS-F-87-958, February 1987.

(Brogi *et al.*, 1990) Brogi, A., Mancarella, P., Pedreschi, D., and Turini, F., Composition Operators for Logic Theories, in *Computational Logic*, J.W. Lloyd (Ed.), 117-134, Springer-Verlag, Berlin, 1990.

(Buntine, 1988) Buntine, W., Generalized Subsumption and Its Applications to Induction and Redundancy, *Artificial Intelligence*, 36, 149-176, 1988.

(Carbonell *et al.*, 1983) Carbonell, J.G., Michalski, R.S., and Mitchell, T.M., An Overview of Machine Learning, in *Machine Learning: An Artificial Intelligence Approach*, R.S. Michalski, J.G. Carbonell and T.M. Mitchell (Eds.), 3-24, Tioga Publishing Company, Palo Alto, CA, 1983.

(Clark, 1978) Clark, K.L., Negation as failure, in *Logic and Databases*, H. Gallaire and J. Minker (Eds.), 293-321, Plenum Press, New York, 1978.

(Datta and Kibler, 1993) Datta, P., and Kibler, D., Concept Sharing: A Means to Improve Multi-Concept Learning, *Proceedings of the Tenth International Conference on Machine Learning*, Amherst, MA, 89-96, 1993.

(De Raedt, 1991) De Raedt, L., *Interactive Concept-Learning*, Ph.D. thesis, Katholieke Universiteit Leuven, 1991.

(Dietterich, 1986) Dietterich, T.G., Learning at the Knowledge Level, *Machine Learning* 1, 3, 287-316, 1986.

(Dietterich & Michalski, 1983) Dietterich, T.G., Michalski, R.S., A Comparative Review of Selected Methods for Learning from Examples, in *Machine Learning: An Artificial Intelligence Approach*, R.S. Michalski, J.G. Carbonell, T.M. Mitchell (Eds.), 41-81, Tioga Publishing Company, Palo Alto, CA, 1983.

(Dzeroski and Bratko, 1992) Dzeroski, S., and Bratko, I., Handling Noise in Inductive Logic Programming, *Proceedings of The Second International Workshop on Inductive Logic Programming ILP 92*, Tokyo, Japan, 1992.

(Esposito, 1990) Esposito, F., Automated acquisition of production rules by empirical supervised learning methods, in *Data, Expert Knowledge and Decisions*, (Vol. II), M. Schader (Ed.), Springer-Verlag, Heidelberg, Germany, 1990.

(Esposito *et al.*, 1992) Esposito, F., Malerba, D., and Semeraro, G., Negation as Specializing Operator, *Proceedings of the Third Italian Workshop on Machine Learning GAA92*, Rome, Italy, May 1992.

(Esposito *et al.*, 1993a) Esposito, F., Malerba, D., Semeraro, G., and Pazzani, M., A Machine Learning Approach To Document Understanding, *Proceedings of the Second International Workshop on Multistrategy Learning MSL-93*, Harpers Ferry, West Virginia, 276-292, 1993.

(Esposito *et al.*, 1993b) Esposito, F., Malerba, D., and Semeraro, G., Specialization in Incremental Learning: The Negation Operator, *Proceedings of the AAAI-93 Spring Symposium Series on "Training Issues in Incremental Learning"*, Stanford, CA, 1993.

(Esposito *et al.*, 1993c) Esposito, F., Malerba, D., and Semeraro, G., Negation as a Specializing Operator, in *Lecture Notes in Artificial Intelligence - Proceedings of the Third Congress of the Italian Association for Artificial Intelligence AI*IA 93*, Turin, Italy, (forthcoming), 1993.

(Esposito *et al.*, 1993d) Esposito, F., Malerba, D., and Semeraro, G., Learning Contextual Rules in First Order Logic, *Proceedings of the Fourth Italian Workshop on Machine Learning GAA93*, Milan, Italy, June 1993.

(Finin, 1986) Finin, T.W., Interactive Classification: A Technique for Acquiring and Maintaining Knowledge Bases, *Proceedings of the IEEE* 74, 10, 1414-1421, 1986.

(Garey and Johnson, 1979) Garey, M.R., and Johnson, D.S., *Computers and Intractability*, Freeman, San Francisco, CA, 1979.

(Gold, 1967) Gold, E.M., Language Identification in the Limit, *Information and Control*, 10, 447-474, 1967.

(Haussler, 1989) Haussler, D., Learning Conjunctive Concepts in Structural Domains, *Machine Learning* 4, 1, 7-40, 1989.

(Hayes-Roth, 1974) Hayes-Roth, F., Schematic classification problems and their solution, *Pattern Recognition*, 105 - 113, 1974.

(Hayes-Roth and McDermott, 1977) Hayes-Roth, F., and McDermott, J., Knowledge acquisition from structural descriptions, *Proceedings of the 5th International Joint Conference on Artificial Intelligence*, Cambridge, MA, 356-362, 1977.

(Helft, 1987) Helft, N., Inductive Generalization: A Logical Framework, in *Progress in Machine Learning - Proceedings of EWSL 87: 2nd European Working Session on Learning*, I. Bratko & N. Lavrac (Eds.), Sigma Press, Bled, Yugoslavia, 149-157, 1987.

(Helft, 1989) Helft, N., Induction as nonmonotonic inference, *Proceedings of the First International Conference on Knowledge Representation and Reasoning*, 1989.

(Horak, 1985) Horak, W., Office Document Architecture and Office Document Interchange Formats: current status of international standardization, *IEEE Computer*, October 1985.

(Idestam-Almquist, 1993) Idestam-Almquist, P., Generalization under Implication by Recursive Anti-unification, *Proceedings of the Tenth International Conference on Machine Learning*, Amherst, MA, 151-158, 1993.

(INTREPID, 1992) Esposito, F., Malerba, D., Semeraro, G., Annese, E., and Ugliano, P., Esprit Project 5203 INTREPID, *Deliverable DAP.1 Document Automatic Classification based on layout analysis*, June 1992.

(Jaffar and Lassez, 1987) Jaffar, J., and Lassez, J.-L., Constraint Logic Programming, *Proceedings of the 14th ACM Symposium on Principles Of Programming Languages POPL-87*, Munich, 111-119, 1987.

(Jung, 1993) Jung, B., On Inverting Generality Relations, *Proceedings of The Third International Workshop on Inductive Logic Programming ILP'93*, Bled, Slovenia, 87-101, 1993.

(KEE User's Manual, 1986) *KEE Software Development System User's Manual*, Intellicorp, 1986.

(Kietz, 1993) Kietz, J.U., A Comparative Study of Structural Most Specific Generalizations Used in Machine Learning, *Proceedings of The Third International Workshop on Inductive Logic Programming ILP'93*, Bled, Slovenia, 149-164, 1993.

(Knight, 1989) Knight, K., Unification: A Multidisciplinary Survey, *ACM Computing Surveys*, Vol.21, No.1, 1989.

(Larson, 1977) Larson, J.B., *Inductive Inference in the Variable Valued Predicate Logic System VL_{21}: Methodology and Computer Implementation*, Ph.D. dissertation, Department of Computer Science, University of Illinois, Urbana, Illinois, May 1977.

(Larson & Michalski, 1975) Larson, J.B., Michalski, R.S., *AQVAL/1 (AQ7) User's Guide and Program Description*, Technical Report, Department of Computer Science, University of Illinois, Urbana, Illinois, UIUCDCS-R-75-731, June 1975.

(Lassez *et al.*, 1988) Lassez, J.-L., Maher, M.J., and Marriott, K., Unification revisited, in *Foundations of Deductive Databases and Logic Programming*, J. Minker (Ed.), 587-625, Morgan Kaufmann, San Mateo, CA, 1988.

(Levesque and Brachman, 1987) Levesque, H.J., and Brachman, R.J., Expressiveness and Tractability in Knowledge Representation and Reasoning, *Computational Intelligence* 3, 2, 78-93, 1987.

(Lloyd, 1987) Lloyd, J.W., *Foundations of Logic Programming*, Second Edition, Springer-Verlag, New York, 1987.

(MacGregor, 1988) MacGregor, R., A Deductive Pattern Matcher, *Proceedings of AAAI-88*, St.Paul, Minnesota, 403-408, 1988.

(Manago, 1986) Manago, M., *Looking for Similarities with MAGGY*, Rapport Interne, LRI, Orsay, 1986.

(Manago and Kodratoff, 1987) Manago, M., and Kodratoff, Y., Model-Driven Learning of Disjunctive Concepts, in *Progress in Machine Learning - Proceedings of EWSL 87: 2nd European Working Session on Learning*, I. Bratko & N. Lavrac (Eds.), Sigma Press, Bled, Yugoslavia, 183-198, 1987.

(Mancarella and Pedreschi, 1988) Mancarella, P., and Pedreschi, D., An Algebra of Logic Programs, *Proceedings of the Fifth International Conference, Symposium of Logic Programming*, Seattle, 1006-1023, 1988.

(Markov, 1993) Markov, Z., Inductive Inference in Networks of Relations, *Proceedings of The Third International Workshop on Inductive Logic Programming ILP'93*, Bled, Slovenia, 265-277, 1993.

(Martelli and Montanari, 1982) Martelli, A., and Montanari, U., An efficient unification algorithm, *ACM Transactions on Programming Languages and Systems*, Vol. 4, No. 2, 258-282, 1982.

(Michalski, 1969) Michalski, R.S., On the quasi-minimal solution of the general covering problem, *Proceedings of the Fifth International Federation on Automatic Control*, 27, 109-129, 1969.

(Michalski, 1980) Michalski, R.S., Pattern Recognition as Rule-Guided Inductive Inference, *IEEE Transactions on Pattern Analysis and Machine Intelligence*, Vol. PAMI-2, 349-361, 1980.

(Michalski, 1983) Michalski, R.S., A Theory and Methodology of Inductive Learning, in *Machine Learning: An Artificial Intelligence Approach*, R.S. Michalski, J.G. Carbonell, T.M. Mitchell (Eds.), 83-134, Tioga Publishing Company, Palo Alto, CA, 1983.

(Michalski and Larson, 1983) Michalski, R.S., and Larson, J.B., *Incremental Generation of VL_1 hypotheses: the underlying methodology and the description of program AQ11*. Revised by K. Chen. Technical Report, Department of Computer Science, University of Illinois at Urbana-Champaign, Urbana, Illinois, UIUCDCS-F-83-905, 1983.

(Mitchell, 1982) Mitchell, T.M., Generalization as Search, *Artificial Intelligence*, 18, 203-226, 1982.

(Muggleton, 1991) Muggleton, S., Inductive Logic Programming, *New Generation Computing*, 8(4), 295-318, 1991 (also in *Inductive Logic Programming*, S. Muggleton (Ed.), 3-27, Academic Press, San Diego, CA, 1992).

(Muggleton and Buntine, 1988) Muggleton, S., and Buntine, W., Machine invention of first-order predicates by inverting resolution, *Proceedings of the Fifth International Conference on Machine Learning*, Ann Arbor, MI, 339-352, 1988 (also in *Inductive Logic Programming*, S. Muggleton (Ed.), 261-280, Academic Press, San Diego, CA, 1992).

(Muggleton and Feng, 1990) Muggleton, S., and Feng, C., Efficient Induction of Logic Programs, *Proceedings of the First Conference on Algorithmic Learning Theory*, Tokio, 1990 (also in *Inductive Logic Programming*, S. Muggleton (Ed.), 281-298, Academic Press, San Diego, CA, 1992).

(Napoli and Ducournau, 1992) Napoli, A., and Ducournau, R., Subsumption in Object-Based Representations, *Proceedings of ERCIM 92*, Pisa, Italy, 1-9, 1992.

(Niblett, 1988) Niblett, T., A study of generalization in Logic Programs, *Proceedings of the Third European Working Session on Learning*, Pitman, London, 131-138, 1988.

(Paterson and Wegman, 1978) Paterson, M.S., and Wegman, M.N., Linear Unification, *J. Comput. Syst. Sci.*, 16, 2, 158-167, 1978.

(Pazzani and Brunk, 1993) Pazzani, M.J., and Brunk, C., *FOCL-1-2-3 Version 1.1*, Department of Information and Computer Science, University of California, Irvine, California, March 1993.

(Pazzani and Kibler, 1992) Pazzani, M.J., and Kibler, D., The utility of knowledge in inductive learning, *Machine Learning* 9, 1, 57-94, 1992.

(Plotkin, 1970) Plotkin, G.D., A Note on Inductive Generalization, in *Machine Intelligence 5*, B. Meltzer and D. Michie (Eds.), 153-163, Edinburgh University Press, 1970.

(Plotkin, 1971a) Plotkin, G.D., A Further Note on Inductive Generalization, in *Machine Intelligence 6*, B. Meltzer and D. Michie (Eds.), 101-124, American Elsevier, New York, 1971.

(Plotkin, 1971b) Plotkin, G.D., *Automatic Methods of Inductive Inference*, Ph.D. thesis, Edinburgh University, 1971.

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

(Quinlan, 1990) Quinlan, J. R., Learning Logical Definitions from Relations, *Machine Learning* 5, 3, 239-266, 1990.

(Quinlan, 1991) Quinlan, J. R., Determinate Literals in Inductive Logic Programming, *Proceedings of the 11th International Joint Conference on Artificial Intelligence*, Sydney, Australia, 746-750, 1991.

(Quinlan and Cameron-Jones, 1993) Quinlan, J. R., Cameron-Jones, R.M., FOIL: a midterm report, in *Lecture Notes in Artificial Intelligence - Proceedings of the Sixth European Conference on Machine Learning ECML-93*, P. Brazdil (Ed.), Springer-Verlag, Berlin, 1993.

(Rissanen, 1983) Rissanen, J., A universal prior for integers and estimation by minimum description length, *Annals of Statistics*, 11, 1, 416-431, 1983.

(Robinson, 1965) Robinson, J. A., A machine-oriented logic based on the resolution principle, *JACM* 12, 1, 23-41, 1965.

(Rouveirol, 1992) Rouveirol, C., Extensions of Inversion of Resolution Applied to Theory Completion, in *Inductive Logic Programming*, S. Muggleton (Ed.), 63-92, Academic Press, San Diego, CA, 1992.

(Rouveirol and Puget, 1990) Rouveirol, C., and Puget, J.-F., Beyond Inversion of Resolution, *Proceedings of the Seventh International Conference on Machine Learning*, Austin, Texas, 122-130, 1990.

(Sanfeliu and Fu, 1983) Sanfeliu, A., and Fu, K.S., A Distance Measure Between Attributed Relational Graphs for Pattern Recognition, *IEEE Transactions on Systems, Man, and Cybernetics*, Vol. SMC-13, No.3, 353-362, 1983.

(Semeraro, 1988) Semeraro, G., *Un Sistema per l'Apprendimento Induttivo Concettuale da Esempi con Logica Variable-Valued*, Laurea dissertation, Università di Bari (also available as LACAM Technical Report, Dipartimento di Informatica, Università di Bari), 1988.

(Shapiro, 1981) Shapiro, E.Y., An Algorithm that Infers Theories from Facts, *Proceedings of the 7th International Joint Conference on Artificial Intelligence*, Vancouver, Canada, 446 - 451, 1981.

(Shapiro *et al.*, 1988) *Encyclopedia of Artificial Intelligence*, E.Y. Shapiro (Ed.), 493-506, 1988.

(Siekmann, 1990) Siekmann, J.H., An Introduction to Unification Theory, in *Formal Techniques in Artificial Intelligence, A Sourcebook*, R.B. Banerji (Ed.), Elsevier Science Publishers B.V. (North Holland), 1990.

(Silverstein and Pazzani, 1991) Silverstein, G., and Pazzani, M., Relational clichés: constraining constructive induction during relational learning, *Proceedings of the Eighth International Workshop on Machine Learning*, Evanston, Illinois, 203-207, 1991.

(Sowa, 1984) Sowa, J.F., *Conceptual Structures: Information Processing in Mind and Machine*, Addison-Wesley, Reading, MA, 1984.

(Stephens, 1990) Stephens, L.M., The Classification of Semantic Relations Based on Primitive Properties, *Proceedings of the First ASIS SIG/CR Classification Research Workshop*, 161-170, 1990.

(Stepp, 1984) Stepp, R. E., *Conjunctive Conceptual Clustering: A Methodology and Experimentation*, Ph.D. dissertation, University of Illinois at Urbana-Champaign, Urbana, Illinois, UIUCDCS-R-84-1189, 1984.

(VanLehn, 1989) VanLehn, K., Efficient Specialization of Relational Concepts, *Machine Learning* 4, 1, 99-106, 1989.

(Vere, 1975) Vere, S.A., Induction of concepts in the predicate calculus, *Proceedings of the 4th International Joint Conference on Artificial Intelligence*, Tbilisi, USSR, 281 - 287, 1975.

(Vere, 1977) Vere, S.A., Induction of relational productions in the presence of background information, *Proceedings of the 5th International Joint Conference on Artificial Intelligence*, Cambridge, MA, 349-355, 1977.

(Vere, 1980) Vere, S.A., Multilevel Counterfactuals for Generalizations of Relational Concepts and Productions, *Artificial Intelligence*, 14, 139-164, 1980.

(Vrain, 1990) Vrain, C., OGUST: A System that Learns using Domain Properties Expressed as Theorems, in *Machine Learning: An Artificial Intelligence Approach*, Y. Kodratoff, R.S. Michalski (Eds.), 360-382, Morgan Kaufmann, San Mateo, CA, 1990.

(Winston, 1970) Winston, P.H., *Learning Structural Descriptions from Examples*, Ph.D. dissertation, Department of Electrical Engineering, Massachussetts Institute of Technology, Cambridge, MA, January 1970.

(Wirth, 1989) Wirth, R., Completing Logic Programs by Inverse Resolution, *Proceedings of the Fourth European Working Session on Learning*, Pitman, London, 239-250, 1989.

(Wirth and O'Rorke, 1991) Wirth, R., and O'Rorke, P., Constraints on Predicate Invention, *Proceedings of the Eighth International Workshop on Machine Learning*, Evanston, Illinois, 457-461, 1991 (also in *Inductive Logic Programming*, S. Muggleton (Ed.), 299-318, Academic Press, San Diego, CA, 1992).

(Wong and You, 1985) Wong, A.K.C., and You, M., Entropy and Distance of Random Graphs with Application to Structural Pattern Recognition, *IEEE Transactions on Pattern Analysis and Machine Intelligence*, Vol. PAMI-7, No.5, 599-609, 1985.

(Woods, 1991) Woods, W.A., Understanding Subsumption and Taxonomy: A Framework for Progress, in *Principles of Semantic Networks: Explorations in the Representation of Knowledge*, J.F. Sowa (Ed.), 45-94, Morgan Kaufmann, San Mateo, CA, 1991.

Appendix A.

The complete descriptions of the five examples for the problem *arch* are:

A_1 arch(obj1) :- cont(obj1,p1), cont(obj1,p2), cont(obj1,p3),
on(p1,p2), on(p1,p3),
touch(p1,p2), touch(p1,p3), touch(p2,p1), touch(p3,p1),
or-hor(p1), or-ver(p2), or-ver(p3),
sh-rect(p1), sh-rect(p2), sh-rect(p3)

A_2 arch(obj2) :- cont(obj2,p4), cont(obj2,p5), cont(obj2,p6),
on(p4,p5), on(p4,p6),
touch(p4,p5), touch(p4,p6), touch(p5,p4), touch(p6,p4),
or-not-appl(p4), or-ver(p5), or-ver(p6),
sh-tri(p4), sh-rect(p5), sh-rect(p6)

N_1 non-arch(obj3) :- cont(obj3,p7), cont(obj3,p8), cont(obj3,p9),
on(p8,p7), on(p9,p7),
touch(p8,p7), touch(p9,p7), touch(p7,p8), touch(p7,p9),
or-hor(p7), or-ver(p8), or-ver(p9),
sh-rect(p7), sh-rect(p8), sh-rect(p9)

N_2 non-arch(obj4) :- cont(obj4,p10), cont(obj4,p11), cont(obj4,p12),
on(p10,p11), on(p10,p12),
touch(p10,p11), touch(p10,p12), touch(p11,p10), touch(p12,p10),
or-hor(p10), or-hor(p11), or-hor(p12),
sh-rect(p10), sh-rect(p11), sh-rect(p12)

N_3 non-arch(obj5) :- cont(obj5,p13), cont(obj5,p14), cont(obj5,p15),
on(p13,p14), on(p13,p15),
touch(p13,p14), touch(p13,p15), touch(p14,p13), touch(p15,p13),
touch(p14,p15),
or-hor(p13), or-ver(p14), or-ver(p15),
sh-rect(p13), sh-rect(p14), sh-rect(p15)