NORTH-HOLLAND

COMPLETENESS AND PROPERNESS OF REFINEMENT OPERATORS IN INDUCTIVE LOGIC PROGRAMMING

PATRICK R. J. VAN DER LAAG AND SHAN-HWEI NIENHUYS-CHENG

▷ Within Inductive Logic Programming, refinement operators compute a set of specializations or generalizations of a clause. They are applied in model inference algorithms to search in a quasi-ordered set for clauses of a logical theory that consistently describes an unknown concept. Ideally, a refinement operator is *locally finite*, *complete*, and *proper*. In this article we show that if an element in a quasi-ordered set $\langle S, \geq \rangle$ has an infinite or incomplete cover set, then an ideal refinement operator for $\langle S, \geq \rangle$ does not exist. We translate the nonexistence conditions to a specific kind of infinite ascending and descending chains and show that these chains exist in unrestricted sets of clauses that are ordered by θ-subsumption. Next we discuss how the restriction to a finite ordered subset can enable the construction of ideal refinement operators. Finally, we define an ideal refinement operator for restricted θ-subsumption ordered sets of clauses. © Elsevier Science Inc., 1998 ◁

1. INTRODUCTION

1.1. Refinement Operators and Ideal Properties

One of the major tasks in Inductive Logic Programming (ILP) is model inference (or concept learning), the induction of logical theories from examples. For a survey of the theory and methods of ILP, we refer to [6]; the foundations of this field are described in [8].

Address correspondence to Patrick R. J. van der Laag, Rabofacet, Teleservices, Datacentrum, P.O. Box 80, 5680 AB Best, The Netherlands; Shan-Hwei Nienhuys-Cheng, Department of Computer Science, Erasmus University of Rotterdam, P.O. Box 1738, 3000 DR Rotterdam, The Netherlands.
Received October 1995; accepted January 1997.

In 1981, Shapiro presented his famous Model Inference System, a milestone in machine learning. In his report [15], he describes how a logical theory for an unknown concept can be inferred by adapting a conjecture (a finite set of clauses) to a sequence of examples of the unknown concept. Starting with a strong conjecture, too general clauses are removed and specializations of removed clauses can be added until a consistent theory is found. These specializations are computed by a *downward refinement operator*. Instead of searching for more specific clauses when necessary, we can also start with a weak conjecture and search for more general clauses that are computed by an *upward refinement operator* [4, 5].

In this article we abstract from the learning systems in which they are used and concentrate on refinement operators. Following Laird [4], we consider downward and upward refinement operators that can be defined on any quasi-ordered set $\langle S, \geq \rangle$. For every C in S, a refinement operator delivers a subset of the set of all elements D of S such that $C \geq D$ (downward) or $D \geq C$ (upward). These elements are called the one-step refinements of C.

Since refinement operators are usually used to search for clauses, most properties of refinement operators for quasi-ordered sets will be illustrated in terms of generalizations and specializations of clauses. If S denotes a set of clauses in a language of first-order logic, then $\geq$ expresses a more-general-than relation between the clauses in S. Whereas logical implication between clauses is the most straightforward notion of generality, θ-subsumption, a weaker version of it, is usually adopted because it is decidable and more efficient for incremental search.

We call a refinement operator *ideal* if it is *locally finite*, *complete*, and *proper*. Local finiteness means that the set of one-step refinements of every clause in the search space is finite and computable. Clearly, refinement operators that are not locally finite are not of any practical use. A refinement operator is complete if (a clause equivalent to) every specialization or generalization of a clause can be connected to it by a finite chain of one-step refinements. When incomplete refinement operators are used, it is not guaranteed that all clauses of a target theory can be derived. If a refinement operator never returns a specialization or generalization that is equivalent to the refined clauses, then it is called *proper*. Properness is a nice property because we are not interested in refinements that are equivalent to a formerly refuted clauses. Moreover, allowing refinements that are equivalent to a formerly refuted clause (as, for example, FOIL [11] does) can cause (infinitely) long chains of equivalent clauses to be generated during the search process.

Our notion of ideality does not capture every desirable aspect of refinement. For example, an ideal refinement operator may return both a proper specialization and a proper specialization of this proper specialization in the same refinement step. Furthermore, the second proper specialization may also be a one-step refinement of the first. This behavior is often regarded as undesirable, because it causes the same clause to be generated many times.

1.2. *Nonexistence Conditions for Ideal Refinement Operators*

To analyze ideal refinement operators carefully, we explore the mathematical concepts that are related to quasi-ordered sets and relevant for ideal refinement operators: covers and infinite ascending and descending chains. If we are searching for the clauses of a logical theory, refinement steps on one hand must be large enough such that every specialization or generalization can be found in finitely many steps. On the other hand, it is important that refinement steps are not too big, because then a target clause might be skipped and a target theory will not be derived. For a refinement operator, idealness implies that the one-step refinements of a clause include all most general proper specializations or most specific proper generalizations. The mathematical concept that resembles this is the concept of *covers*. A natural way to define refinement operators then is to assign to each clause its downward or upward *cover set*. We can, for example, use Reynolds' [13] cover relation between atoms to define ideal downward and upward refinement operators for substitution ordered sets of atoms. For clauses, however, this approach has some problems:

1. In practical generality orderings like θ-subsumption and logical implication, it is hard to test whether a clause covers another clause. Generating downward or upward covers constructively is even harder.
2. A cover set can be incomplete. We will give an example of a clause that has infinitely many nonequivalent proper generalizations but has an empty upward cover set.
3. A cover set can contain infinitely many nonequivalent clauses.
4. Even when every clause is known to have a finite and complete downward or upward cover set, completeness of refinement operators that return such cover sets is not guaranteed. Moreover, we will describe an abstract ordered set in which every element has a finite and complete cover set, but for which a complete refinement operator does not exist.

Despite all these problems, the concept of cover sets is important for ideal refinement operators. As we will prove in Section 3, an ideal refinement operator always returns a superset of a finite and complete cover set. Consequently, if an element of S has an incomplete or infinite cover set in $\langle S, \geq \rangle$, then this implies *nonexistence* of an ideal refinement operator for $\langle S, \geq \rangle$. These nonexistence conditions will be translated into so-called *uncovered infinite ascending and descending chains*. The related theorems will be applied to $\langle \mathscr{C}, \succeq \rangle$, where $\mathscr{C}$ refers to the set of all clauses in a first-order language and $\succeq$ refers to θ-subsumption. We will provide uncovered chains for concrete clauses to show the nonexistence of both ideal upward and downward refinement operators for $\langle \mathscr{C}, \succeq \rangle$.

1.3. Ideal Refinement Operators for Finite Quasi-Ordered Sets

If we have a *finite* quasi-ordered set $\langle S, \geq \rangle$, and $C \geq D$ tests are decidable for all elements of S, then we can theoretically define ideal refinement operators using cover sets as a set of one-step refinements. However, to extract cover sets from S, many tests are required, which does not yield efficient refinement operators. In practice, refinements should be derivable from the refined element in a constructive way. For example, Reynolds [13] has described simple and constructive downward refinement operations for atoms. Each of these operations is a simple substitution, for example, the unification of two variables. Shapiro [15] combined these simple substitutions in a downward refinement operator that constructs downward cover sets of atoms. This ideal refinement operator for atoms has been generalized to an ideal refinement operator for clauses with respect to the so-called *substitution ordering* [9]. When these simple substitutions are applied to clauses that are ordered by θ-subsumption, however, then the resulting clauses are sometimes equivalent to the refined clause. To define an ideal refinement operator, these improper refinements should be avoided. For a complete refinement operator for θ-subsumption ordered clauses, we must also introduce an operation by which we can add one or more literals. In Section 4, we describe how *inverse reduction* is related to proper substitutions and the addition of one or more literals. The applicability of substitutions, literal additions, and inverse reduction is also influenced by the way in which the search space of clauses is restricted to a finite set. We will restrict sets of *reduced* clauses $\mathscr{R}$ to finite sets $\mathscr{R}^{newsize}$, using our complexity measure *newsize*. In the end we will discuss our downward refinement operator ρ_r, which is ideal for quasi-ordered sets $\langle \mathscr{R}^{newsize}, \succeq \rangle$.

1.4. Related Work

Shapiro [15] has defined a locally finite and proper downward refinement operator for finite sets of reduced clauses (ρ_0). This refinement operator returns proper specializations only, and no one-step refinement of a clause C is a proper specialization of another one-step refinement of C. ρ_0 was probably intended to compute downward cover sets. It was certainly intended to be what we call *weakly complete*, but, as is shown in [1] and [7], it is not. In the first of these articles we have first presented our inverse reduction algorithm, our complexity measure *newsize*, and our downward refinement operator ρ_r.

In another article [3], we have proved the nonexistence of ideal upward and downward refinement operators for unrestricted θ-subsumption ordered sets of clauses $\langle \mathscr{C}, \succeq \rangle$ and defined a locally finite, complete but *improper* upward refinement operator for such sets. This upward refinement operator is a counterpart of Laird's [4] downward refinement operator that is also locally finite, complete, and improper for $\langle \mathscr{C}, \succeq \rangle$. In [2], the nonexistence conditions for ideal refinement operators have been generalized by using cover sets and uncovered infinite chains, and, using a logical framework of refinement operators, different approaches to approximate ideal refinement operators have been discussed.

In the article [9] we have discussed five subsequent weakenings of logical implication, and we have defined downward and upward refinement operators for a finite set of clauses with respect to all of these orderings. Among the described refinement operators are ρ_r and its upward dual, which is an ideal upward refinement operator for finite θ-subsumption ordered sets $\langle \mathscr{R}^{newsize}, \succeq \rangle$.

Ling and Dawes [5] have also defined an upward refinement operator for finite θ-subsumption ordered sets of clauses. This refinement operator is neither complete nor proper. However, considering a variant of θ-subsumption where different variables always refer to different objects, Ling and Dawes' upward refinement operator for clauses becomes ideal. In this ordering (called θ-subsumption under object identity [14]), $q \leftarrow p(X,Y), p(Y,X)$ is not regarded as more general than $q \leftarrow p(X,X)$, and hence cannot be derived from it.

In the case of downward refinement operators, our notion of idealness can be contrasted with the notion of *optimality* [6]. Using an optimal downward refinement operator, no clause in the search space can be generated more than once, which makes optimal refinement very efficient. Optimal refinement is particularly interesting when the search starts with the top element (e.g., the empty clause), in which case weaker notions of completeness are sufficient to guarantee the derivability of all clauses in the search space. An example of such a weak notion of completeness is *global completeness* [6], which means that there exists a chain of refinements from the top element of the search space (e.g., the empty clause) to every other clause, Van Laer [16] has defined a globally complete optimal downward refinement operator for sets of function-free, range-restricted clauses that are ordered by θ'-subsumption (C θ'-subsumes D iff there exists some variable renaming θ for which $C\theta \subseteq D$).

So far, globally complete optimal downward refinement operators for θ-subsumption ordered sets have not been defined. Moreover, if one-step refinements must be covers (which is often included in the definition of refinement operators, e.g., in [12] and [6]), then such an optimal refinement operator does not exist, as will be shown at the end of Section 3.2.

In all cases of upward refinement and when downward refinement does not start with the top element of the search space, global completeness and optimality are less appropriate. In these cases *local completeness* [6] can be favored over global completeness, and our notion of idealness can be favored over optimality.

2. NOTATION AND DEFINITIONS

2.1. *Quasi-Ordered Set*

Let S be a set and let C, D, E be elements of S.

- A binary relation $\geq$ on S is called a *quasi-ordering* on S if it is reflexive ($C \geq C$) and transitive ($C \geq D$ and $D \geq E$ imply $C \geq E$). We use $C > D$ to denote $C \geq D$ and $D \not\geq C$. For every quasi-ordering $\geq$ we can define an *equivalence* relation $\sim$ by $C \sim D$, if $C \geq D$ and $D \geq C$.

2.2. *Refinement Operators*

Given a quasi-ordered set $\langle S, \geq \rangle$, and $C, D, E \in S$:

- If $C \geq D$ or $D \geq C$, then C and D are called *comparable.*
- ρ is called a *downward refinement operator* if $\forall C \in S$: $\rho(C) \subseteq \{D \in S | C \geq D\}$.
- δ is called an *upward refinement operator* if $\forall C \in S$: $\delta(C) \subseteq \{D \in S | D \geq C\}$.
- Given ρ (or δ), the sets of *one-step refinements*, *n-step refinements*, and *refinements* of C are, respectively,

$$\rho^1(C) = \rho(C)$$

$$\rho^n(C) = \{D | \exists E \in \rho^{n-1}(C) \quad \text{and} \quad D \in \rho(E)\}$$

$$\rho^*(C) = \rho^1(C) \cup \rho^2(C) \cup \cdots \cup \rho^i(C) \cup \cdots$$

- Every sequence $C = C_0, C_1, \ldots, C_n = D$ such that $C_i \in \rho(C_{i-1})$ is called a *ρ-chain* C to D.

2.3. *Ideal Properties*

Given a quasi-ordered set $\langle S, \geq \rangle$, and $C, D, E \in S$, ρ (and dually δ) is called

- *locally finite* if $\forall C \in S$: $\rho(C)$ is finite and computable
- *complete* if $\forall C > D$: $\exists E \in \rho^*(C)$ such that $E \sim D$
- *proper* if $\forall C \in S$: $\rho(C) \subseteq \{D \in S | C > D\}$
- *ideal* if ρ is locally finite, complete and proper.

2.4. *Cover Sets*

Given a quasi-ordered set $\langle S, \geq \rangle$, and $C, D, E \in S$:

- *C covers D* if $C > D$ and $\nexists E$: $C > E > D$.
- If C covers D, then we call C an *upward cover* of D and D a *downward cover* of C.
- *Downward* and *upward cover sets* are maximum sets of nonequivalent downward and upward covers, respectively. They are denoted by $dc(C)$ and $uc(C)$.
- $dc(C)$ is called *complete* if for every $D \in S$ such that $C > D$, $\exists E \in dc(C)$ such that $E \geq D$. Complete upward cover sets are defined dually.

Note that cover sets are only uniquely up to equivalence. When an element in a cover set is replaced by an equivalent element, the result is also a cover set.

Ordered sets with incomplete cover sets are common in mathematics. For example, consider the set of real numbers ordered in the usual way; then there is no pair of numbers X, Y such that $X > Y$ and Y is a downward cover of X.

2.5. *Clauses*

Given a language of first-order logic $\mathscr{L}$ with finitely many function, constant, and predicate symbols, we use the following notation. Function symbols are denoted by f, g; constants by a, b; variables by X, Y, Z; predicate symbols by p, q, r; literals by L, M; clauses by $C, D, \ldots$; and sets of clauses by S. All of these symbols can occur with subscripts.

A clause represents a (possibly empty) set of literals. The empty clause is denoted by $\square$. By $C \backslash D$ we denote the difference set of C and D that is obtained by removing all literals in D from C.

Usually, clauses will be written in the usual logic programming style. For example, the clause

$$\{p(X,X), \neg q(X,Y), \neg q(Y,X)\}$$

can be written as

$$p(X,X) \leftarrow q(X,Y), q(Y,X).$$

Sometimes these notation styles will be mixed for convenience. Thus the clause

$$D_n = q \leftarrow \left\{p(X_i, X_j) \middle| 1 \leq i, j \leq n, i \neq j\right\}, n \geq 2$$

represents the set of literals $\{q\} \cup \{\neg p(X_i, X_j) | 1 \leq i, j \leq n, i \neq j\}, n \geq 2$.

2.6. θ-Subsumption and Reduction

Given a quasi-ordered set of clauses $\langle S, \geq \rangle$, and $C, D, E \in S$:

- If $C \geq D$ holds, then C is called a *generalization* of D and D a *specialization* of C. If $C > D$, then generalizations and specializations are called *proper*.
- Clause C *θ-subsumes* clause D, denoted by $C \succeq D$, if $C\theta \subseteq D$ for some substitution θ.
- A clause C is called *reduced* iff $D \subseteq C$ and $D \sim C$ imply $C = D$. In words, C is reduced iff it is equivalent to no proper subset of itself.
- If D is a nonreduced clause and C is a reduced clause such that $C \subset D$ and $C \sim D$, then all literals in $D \setminus C$ are called *redundant*.

For a given set of clauses, S, θ-subsumption is a quasi-ordering on S. If we talk about a quasi-ordering on the set of all clauses in a first-order language $\mathscr{L}$, then we use $\mathscr{C}$ to denote this *unrestricted* set of clauses. The set of all reduced clauses in a first-order language $\mathscr{L}$ is denoted by $\mathscr{R}$. Given a complexity measure for clause *size* and some fixed upper bound for this complexity measure k, we denote the subsets of $\mathscr{C}$ and $\mathscr{R}$ that contain all (reduced) clauses with a $size \leq k$ by $\mathscr{C}^{size}$ and $\mathscr{R}^{size}$. Throughout this article, clauses that differ only in variable names will be regarded as the same.

3. NONEXISTENCE OF IDEAL REFINEMENT OPERATORS

3.1. Nonexistence Conditions

In this section we relate incomplete or infinite cover sets to the nonexistence of ideal refinement operators for a quasi-ordered set $\langle S, \geq \rangle$. All lemmas in this section will be in terms of downward refinement operators, but will hold similarly for upward refinement operators.

Lemma 3.1. Let ρ be an ideal downward refinement operator for $\langle S, \geq \rangle$; then every element in S has a finite and complete downward cover set.

PROOF. Let ρ be an ideal refinement operator. For all $C \in S$, the following algorithm finds a subset dcc of $\rho(C)$ that is a finite and complete downward cover set of C.

```
$dcc := \rho(C)$
while $\exists D, E \in dcc$ such that $D \neq E$ and $D \geq E$ do
    $dcc := dcc \setminus \{E\}$.
```

Since ρ is locally finite, the algorithm terminates. From the completeness and properness of ρ and the relation between D and E when E is removed from dcc, we can deduce that after termination of the algorithm,

$$\nexists D, E \in dcc \text{ such that } D \neq E \text{ and } D \geq E \tag{1}$$

$$\forall E \in S\colon C > E \text{ implies } \exists D \in dcc \text{ such that } C > D \geq E. \tag{2}$$

First we prove that every clause in dcc is a downward cover of C: Assume that $E \in dcc$ is not a downward cover of C. Then $\exists F \in S$, such that $C > F > E$, and by (2), $\exists D \in dcc$ such that $C > D \geq F$. But then $D, E \in dcc$ and $D > E$, which contradicts (1). Next, since ρ is locally finite, dcc is a finite cover set. By (1), no two elements of dcc are equivalent, and finally, by (2), dcc is complete. Notice that, in the proof above, to compute a complete cover set, it is required that $D \geq E$ tests be decidable. To prove the existence or nonexistence of such a set, however, $\geq$ need not be decidable. □

Corollary 3.1. Given a quasi-ordered set $\langle S, \geq \rangle$, if for some C in S, $dc(C)$ is incomplete or infinite, then an ideal downward refinement operator for $\langle S, \geq \rangle$ does not exist.

At the end of this section we will give an example of a quasi-ordered set in which the cover set of every element is finite and complete, but an ideal refinement operator still does not exist. Although this might not be clear at first, incomplete cover sets are closely related to infinitely ascending or descending chains. It is well known that infinitely ascending and descending chains (infinite chains of proper generalizations or specializations) exist for θ-subsumption ordered clauses, which can cause termination problems in search algorithms (e.g., see [10]). The existence of such infinite chains in a quasi-ordered set alone, however, does not imply that ideal refinement operators cannot exist. Existence of the following, more specific kind of infinite chains does imply the nonexistence of ideal refinement operators.

3.1.1. Uncovered Chains. Given $\langle S, \geq \rangle$ and $C, D_1, D_2, D_3, \ldots \in S$,

- If $D_1 > D_2 > D_3 > \cdots > D_n > D_{n+1} > \cdots > C$, and C has no upward cover $E \in S$ such that $D_n \geq E > C$ for all $n \geq 1$, then $D_1, D_2, D_3, \ldots$ is called an *uncovered infinite descending chain* of C.
- If $C > \cdots > D_{n+1} > D_n > \cdots > D_3 > D_2 > D_1$, and C has no downward cover $E \in S$ such that $C > E \geq D_n$ for all $n \geq 1$, then $D_1, D_2, D_3, \ldots$ is called an *uncovered infinite ascending chain* of C.

The following lemma relates uncovered chains to cover sets and thus, using Lemma 3.1, to the nonexistence of ideal refinement operators.

Lemma 3.2. Given $\langle S, \geq \rangle$, C in S and the following statements:

1. *$dc(C)$ is incomplete,*
2. *C has an uncovered infinite ascending chain,*
3. *$dc(C)$ is infinite or incomplete,*

then 1 *implies* 2 *and* 2 *implies* 3.

PROOF. ($1 \rightarrow 2$) Let $dc(C)$ be incomplete. Then for some $D_1 \in S$ for which $C > D_1$, there exists no $E \in dc(C)$ such that $C > E \geq D_1$. D_1 is not a downward cover of C, otherwise D_1 or another element in S that is equivalent to D_1 would be in $dc(C)$. Hence, there must be an element $D_2 \in S$ such that $C > D_2 > D_1$ and $\nexists E \in dc(C)$ such that $C > E \geq D_2$ (the existence of such an E would also imply $C > E > D_1$). This line of reasoning can be repeated forever; hence C has an uncovered infinite ascending chain $C > \cdots > D_{n+1} > D_n > \cdots > D_2 > D_1$.

($2 \rightarrow 3$) By contradiction. Let $C > \cdots > D_{n+1} > D_n > \cdots > D_2 > D_1$ be an uncovered infinite ascending chain of C. Suppose $dc(C) = \{E_1, E_2, \ldots, E_m\}$ is finite and complete. Completeness of $dc(C)$ implies that for every D_i there exists an E_j

such that $C > E_j \geq D_i$. Since there are finitely many E_j's and infinitely many D_i's, some E_k must satisfy $C > E_k \geq D_i$ for infinitely many D_i's. Thus for every i we can find a $j > i$ such that $E_k \geq D_j$. But then also, $C > E_k \geq D_i$ for all i. This contradicts the statement that $C > \cdots > D_{n+1} > D_n > \cdots > D_2 > D_1$ is an uncovered infinite ascending chain of C.

Corollary 3.2. Given $\langle S, \geq \rangle$, if some C in S has an uncovered infinite ascending chain, then an ideal downward refinement operator for $\langle S, \geq \rangle$ does not exist.

A natural question that remains is whether the existence of finite and complete cover sets for all $C \in S$ implies that $\rho(C) = dc(C)$ is an ideal downward refinement operator. The answer is no, as illustrated by the following example.

Example. Let $S = \{C_1, C_2, \ldots, D\}$ with infinitely many C_i's between C_1 and D, and let the elements of S be ordered as follows:

$$C_1 > C_2 > \cdots > C_n > C_{n+1} > \cdots > D.$$

In the downward case, $dc(C_i) = \{C_{i+1}\}$ and $dc(D) = \{\ \}$; thus every downward cover set is finite and complete. If we now define $\rho(C) = dc(C)$, then ρ is incomplete and hence not ideal: D cannot be derived in finitely many steps from any C_i.

In this ordered search space $\langle S, \geq \rangle$, however, it is still possible to define an ideal downward refinement operator by extending the cover sets:

$$\rho(C_i) = \{C_{i+1}, D\}.$$

For the upward case consider D. Since every C_i in S satisfies $C_i > C_{i+1} > D$, no C_i is an upward cover of D. Hence, $uc(D) = \{\ \}$ and $uc(D)$ is incomplete. $C_1, C_2, \ldots$ is an uncovered infinite ascending chain of D; hence, by the dual of Lemma 3.2, an ideal upward refinement operator for $\langle S, \geq \rangle$ does not exist.

In the example above, the refinement operator $\rho(C) = dc(C)$ could be extended to an ideal refinement operator. The following example shows an ordered search space for which such an extension is not possible. Moreover, it shows that, although all downward cover sets are finite and complete, an ideal downward refinement operator does not exist.

Example. Let $\langle T, \geq \rangle$ be a partially ordered set that contains an infinite binary tree that is rooted by C_0:

$$C_0 \in T, \text{ and if } C_{i_1,\ldots,i_n} \in T, \text{ then also } C_{i_1,\ldots,i_n,0} \in T \text{ and } C_{i_1,\ldots,i_n,1} \in T$$

$$C_{i_1,\ldots,i_n} > C_{i_1,\ldots,i_n,0}$$

$$C_{i_1,\ldots,i_n} > C_{i_1,\ldots,i_n,1}.$$

Thus, there are infinitely many paths of infinite length from C_0 to smaller ($>$) elements. We now define B, the set of "bottoms" as follows. For every infinite path $p = C_0 > C_{0,i_2} > C_{0,i_2,i_3} > \cdots$, there is a D_p in B such that

$$C_0 > C_{0,i_2} > C_{0,i_2,i_3} > \cdots > D_p,$$

and such that D_p is incomparable with every element of T that is not in p. Furthermore, if $p \neq p'$, then D_p and $D_{p'}$ are incomparable.

Let $S = T \cup B$ and let $\geq$ be defined as above. Then every element of S has a finite and complete downward cover set ($dc(C_{i_1,\ldots,i_n}) = \{C_{i_1,\ldots,i_n,0}, C_{i_1,\ldots,i_n,1}\}$, and for every infinite path p, $dc(D_p) = \{\ \}$). Assume that ρ is an ideal downward refinement operator for $\langle S, \geq \rangle$. Then for every D_p, there must be a finite ρ-chain from C to D_p, and, since the elements of B are mutually incomparable, $D_p \in \rho(C)$ must hold for some element C of S. We conclude that, if ρ is ideal for $\langle S, \geq \rangle$, then

$$B \subseteq \bigcup_{C \in T} \rho(C).$$

However, T is a countable set and, since ρ is locally finite, $\bigcup_{C \in T} \rho(C)$ is countable as well. On the other hand, the number of paths starting in C ($\{0,1\}^{\mathbf{N}}$, the number of real numbers) is uncountable and hence B is not a countable set. From this contradiction we conclude that an ideal refinement operator for $\langle S, \geq \rangle$ does not exist. □

3.2. Nonexistence of Ideal Upward Refinement Operators

In this section we will apply the dual of Lemma 3.2 to $\langle \mathscr{C}, \succeq \rangle$, i.e., to unrestricted sets of clauses that are ordered by θ-subsumption, the most widely used notion of generality in model inference. We will show that uncovered descending chains exist for one specific clause, but we can prove that this clause does not form an isolated, exceptional case; uncovered chains exist for infinitely many clauses. Hence, there seems to be no way to find complete refinement operators by treating these exceptional clauses separately.

We will prove that $D_2, D_3, D_4, \ldots$ is an uncovered infinite descending chain of C, where

$$C = q \leftarrow p(X_1, X_1)$$

$$D_n = q \leftarrow \{p(X_i, X_j) | 1 \leq i, j \leq n, i \neq j\} \ (n \geq 2).$$

Clearly, the heads of C, of every D_n, and of every E for which $D_n \succeq E \succeq C$ holds are equal. Since these heads do not affect the derivability of C from any D_n, we concentrate on the bodies of these clauses:

$$K = \{p(X_1, X_1)\}$$

$$K_n = \{p(X_i, X_j) | 1 \leq i, j \leq n, i \neq j\} \ (n \geq 2).$$

Thus K_n represents a structure that is known as a complete graph of size n; for example,

$$K_3 = \{p(X_1, X_2), p(X_1, X_3), p(X_2, X_1), p(X_2, X_3), p(X_3, X_1), p(X_3, X_2)\}.$$

Lemma 3.3. For all $n \geq 2$, K_n is reduced.

PROOF. Assume that K_n is not reduced for some n. Then for some substitution θ, $K_n\theta \subset K_n$. This implies that two literals $p(X_{i_1}, X_{i_2})$ and $p(X_{j_1}, X_{j_2})$ in K_n are mapped to the same literal $p(X_{k_1}, X_{k_2})$ in $K_n\theta$. If $i_1 \neq j_1$, then $p(X_{i_1}, X_{j_1})$ in K_n is mapped to $p(X_{k_1}, X_{k_1})$. Otherwise, $i_2 \neq j_2$ and $p(X_{i_2}, X_{j_2})$ in K_n is mapped to $p(X_{k_2}, X_{k_2})$. Both cases contradict $K_n\theta \subset K_n$. □

Lemma 3.4. $K_2 \succ K_3 \succ \cdots \succ K_n \succ K_{n+1} \succ \cdots \succ K$.

Proof. For every K_n we can define a θ that maps every variable X_i in K_n to X_1. This gives $K_n\theta \subseteq K$. Since $p(X_1, X_1)$ in K cannot be mapped to any literal in any K_n, we get $K_n \succ K$. Using the trivial substitution, we can prove $K_n \succeq K_{n+1}$. Since K_{n+1} is reduced (Lemma 3.3) and $K_n \subset K_{n+1}$, K_{n+1} and K_n cannot be equivalent, and $K_n \succ K_{n+1}$. □

Lemma 3.5. There is no E such that for all $n \geq 2$, $K_n \succeq E \succ K$.

Proof. Assume that some E does satisfy $K_n \succeq E \succ K$ for all $n \geq 2$. Let $X_1, \ldots, X_m$ be all variables in E. Since $E \succ K$, $E\theta \subseteq K$ for some θ and E can contain only literals of the form $p(X_i, X_j)$. In these literals $X_i \neq X_j$ must hold; otherwise E is equivalent to K. But then $E \subseteq K_m$ implies $E \succeq K_m \succ K_{m+1}$, which contradicts $K_{m+1} \succeq E$. □

Corollary 3.3. Let $\langle \mathscr{C}, \succeq \rangle$ be an unrestricted set of clauses ordered by θ-subsumption and let C and $D_2, D_3, D_4, \ldots,$ as defined at the beginning of Section 3.2, be clauses in $\mathscr{C}$. Then $D_2, D_3, D_4, \ldots$ is an uncovered infinite descending chain of C.

Remark. Uncovered infinite descending chains exist for clauses of arbitrary complexity. For example, if we replace all variables X_i in the definition of K and K_n by $f^n(X_i)$, then the proofs of Lemmas 3.3, 3.4, and 3.5 can remain unchanged. Other clauses that can be proved analogously to have an uncovered infinite descending chain are $q \leftarrow p(g(X_1, X_1))$ and $q \leftarrow p(f^n(g(X_1, X_1)))$. Note that these last clauses contain a binary function symbol, but no binary predicate symbols.

Using the same line of reasoning as in the proof of Lemma 3.5, we can verify that no clause E is an upward cover of K. Hence K has infinitely many proper generalizations $K_1, K_2, \ldots,$ but its upward cover set is empty. Similarly, we can prove that $\leftarrow p(X_1, X_1)$ is the only upward cover of $C = q \leftarrow p(X_1, X_1)$. Consequently, if we consider only definite Horn clauses, then C also has infinitely many proper generalizations $D_1, D_2, \ldots,$ and the upward cover set of C in the set of all of definite Horn clauses is empty as well. For both search spaces, the existence of a clause without upward covers implies the nonexistence of an optimal downward refinement operator in the framework of De Readt and Bruynooghe [12]. Adopting their definitions, an optimal downward refinement operator returns only downward covers, and every clause can be derived from the empty clause through exactly one chain of refinements. Since a clause without upward covers is not a downward cover of any other clause, it can never be derived. Hence such an optimal downward refinement operator for the mentioned search spaces does not exist, which is a derived result. The main result for *ideal* upward refinement is the following:

Theorem 3.1. Let $\langle \mathscr{C}, \succeq \rangle$ be a θ-subsumption ordered set, containing all clauses in a first-order language that contains at least one predicate or function symbol of arity ≥ 2. Then an ideal upward refinement operator for $\langle \mathscr{C}, \succeq \rangle$ does not exist.

Proof. Follows directly from the dual of Lemma 3.2 and Corollary 3.3. □

Remark. In [2], we have shown that the infinite chains in this subsection are not only uncovered infinite descending chains w.r.t. θ-subsumption, but also infinite descending chains w.r.t. logical implication. Hence the theorem above is also valid for unrestricted sets of clauses that are ordered by logical implication.

3.3. Nonexistence of Ideal Downward Refinement Operators

In this section we show the nonexistence of ideal *downward* refinement operators for $\langle \mathscr{C}, \succeq \rangle$. Using $C_n = \{p(Y_1,Y_2), p(Y_2,Y_3), \ldots, p(Y_{n-1},Y_n), p(Y_n,Y_1)\}$ ($n \geq 2$), we define the following clauses:

$$C = q \leftarrow p(X_1,X_2), p(X_2,X_1)$$

$$D_n = q \leftarrow \{p(X_1,X_2), p(X_2,X_1)\} \cup C_{3^n} \ (n \geq 1).$$

We will prove that $D_1, D_2, D_3, \ldots$ is an uncovered infinite ascending chain of C. Again, the heads of C, of every D_n, and of every clause E for which $C \succeq E \succeq D_n$ holds do not affect the derivability of C from any D_n, and we will again concentrate on the bodies of these clauses:

$$G = \{p(X_1,X_2), p(X_2,X_1)\}$$

$$G_n = G \cup C_{3^n} \ (n \geq 1).$$

To prove that G and every G_n is reduced, we introduce the notion of *cycles*.

3.3.1. Cycles. A sequence of literals $L_1, \ldots, L_n$ is called a *linked chain* iff L_i and L_{i+1} share at least one variable ($1 \leq i \leq n-1$). Let C be a clause. Then C contains a *cycle* of length n if C contains n distinct literals $L_1, L_2, \ldots, L_n$ ($n \geq 2$), such that the sequence $L_1, L_2, \ldots, L_n, L_1$ is a linked chain.

Example. Every C_n ($n \geq 2$) contains cycles of length n, for example,

$$p(X_2,X_3), p(X_3,X_4), \ldots, p(X_{n-1}), p(X_n), p(X_n,X_1), p(X_1,X_2),$$

but no proper subset of C_n contains a cycle.

The observations in the example above form a part of the proof of the following lemma.

Lemma 3.6. For all $n \geq 2$, C_n is reduced.

PROOF. Assume that C_n is not reduced for some n. Then for some θ, $C_n\theta \subset C_n$. This relation implies that θ maps two literals in C_n to the same literal and hence that θ maps two variables X_i and X_j, $i < j$, to the same variable X_k. If $j = i+1$, then $p(X_i, X_{i+1})\theta = p(X_k, X_k) \notin C_n$; hence $j > i+1$. If θ maps two variables X_i and X_j such that $j > i+1$ to the same variable, then $C_n\theta$ contains a cycle of length $j - i$:

$$p(X_i,X_{i+1})\theta, p(X_{i+1},X_{i+2})\theta, \ldots, p(X_{j-1},X_j)\theta.$$

Since no proper subset of C_n contains a cycle and $C_n\theta$ does, we conclude that $C_n\theta \subset C_n$ cannot hold, and hence that C_n is reduced. □

Lemma 3.7. For all $n \geq 1$, G_n is reduced.

PROOF. Assume that G_n is not reduced for some $n \geq 1$. Then for some θ, $G_n\theta \subset G_n$, and, by definition of G_n, $(G \cup C_{3^n})\theta \subset G \cup C_{3^n}$. Since G (an alphabetical variant of C_2) and C_{3^n} are both reduced (Lemma 3.6), $G\theta \subset G$ or $C_{3^n}\theta \subset C_{3^n}$ cannot hold. Hence θ must map at least one literal of G to a literal of C_{3^n}, or the other way around.

First assume that for some k, $1 \leq k < 3^n$, $p(X_1, X_2)\theta = p(Y_k, Y_{k+1})$. Then $p(X_2, X_1)\theta = p(Y_{k+1}, Y_k)$, but $p(Y_{k+1}, Y_k) \notin G_n$. The same holds when we start with $p(X_2, X_1)$ or when a literal of the G-part is mapped to $p(Y_{3^n}, Y_1)$.

Next we prove the case in which $p(Y_1, Y_2)\theta = p(X_1, X_2)$. Then $p(Y_2, Y_3)\theta = p(X_2, Y_3\theta) \in G_n$ and θ must map Y_3 to X_1. Then also, $p(Y_3, Y_4)\theta = p(X_1, Y_4\theta) \in C_{3^n}$, and θ must map Y_4 to X_2. This line of reasoning can be repeated, and every Y_i with odd index must be mapped to X_1, and every Y_i with an even index must be mapped to X_2. But then, since 3^n is odd, $p(Y_{3^n}, Y_1)\theta = p(X_1, X_1)$, which contradicts $G_n\theta \subset G_n$. The same argument holds when we start with another $p(Y_i, Y_{i+1})$ or $p(Y_{3^n}, Y_1)$ instead of $p(Y_1, Y_2)$, and with $p(X_2, X_1)$ instead of $p(X_1, X_2)$. □

Lemma 3.8. $G \succ \cdots \succ G_{n+1} \succ G_n \succ \cdots \succ G_2 \succ G_1$.

Proof. $G \succ G_n$ follows directly from $G \subset G_n$ and the reducedness of G_n (Lemma 3.7). For all $n \geq 2$, let θ_{n+1} be a substitution that maps every Y_i $(1 \leq i \leq 3^{n+1})$ in G_{n+1} to Y_j in G_n, where $j = 3^n$ iff $i \bmod 3^n = 0$ and $j = i \bmod 3^n$ otherwise. Then $G_{n+1}\theta_{n+1} = G_n$, and hence $G_{n+1} \succeq G_n$. Assume that $G_n \succeq G_{n+1}$ also holds; then for some σ, $G_n\sigma \subseteq G_{n+1}$. But then $(G_{n+1}\theta)\sigma = G_n\sigma \subseteq G_{n+1}$. Together with $|G_n| < |G_{n+1}|$, this implies that $G_{n+1}\sigma\theta \subset G_{n+1}$, which contradicts that G_{n+1} is reduced (Lemma 3.7). We conclude that $G_n \not\succeq G_{n+1}$, and hence $G_{n+1} \succ G_n$ for all $n \geq 1$.

□

Lemma 3.9. There is no E such that for all $n \geq 1$, $G \succ E \succeq G_n$.

Proof. Assume that E is a clause that satisfies $G \succ E \succeq G_n$ for all $n \geq 1$. Choose an m such that $3^m > |E|$. Since $E \succ G_m$, $E\theta \subseteq G_m$ for some θ. Since $|E\theta| < 3^m$ and $|G_m| = 3^m + 2$, we know that at least one of the literals of the C_{3^m}-part of G_m does not occur in $E\theta$. Without loss of generality, we may assume that $p(Y_n, Y_1) \in (G_m \setminus E\theta)$.

Consider the clause $F = G_m \setminus \{p(Y_n, Y_1)\}$. Then $E\theta \subseteq F$ implies $E \succeq F$. Let σ map every Y_i in F to X_1 if i is odd, and to X_2 if i is even. Then $F\sigma \subseteq G$ and hence $F \succeq G$. So $E \succeq F \succeq G$, which contradicts $G \succ E$. □

Corollary 3.4. Let $\langle \mathscr{C}, \succeq \rangle$ be a θ-subsumption ordered set of clauses and let C and let $D_1, D_2, D_3, \ldots,$ as defined at the beginning of Section 3.3, *be clauses in $\mathscr{C}$. Then $D_1, D_2, D_3, \ldots$ is an uncovered infinite ascending chain of C.*

Remark. This can be generalized too. Uncovered infinite ascending chains can be constructed for clauses of arbitrary complexity, using modifications to C and D_n similar to the ones discussed in the remarks following Corollary 3.3.

Theorem 3.2. Let $\langle \mathscr{C}, \succeq \rangle$ be a θ-subsumption ordered set, containing all clauses in a first-order language that contains at least one predicate or function symbol of arity ≥ 2. Then an ideal downward refinement operator for $\langle \mathscr{C}, \succeq \rangle$ does not exist.

Remark. In [2], we have shown that the infinite chains in this subsection are not only uncovered infinite ascending chains w.r.t. θ-subsumption, but also infinite ascending chains w.r.t. logical implication. Hence the theorem above is also valid for unrestricted sets of clauses that are ordered by logical implication.

4. RESTRICTING THE SEARCH SPACE

In the previous sections we have shown that ideal (locally finite, complete, and proper) refinement operators for general unrestricted θ-subsumption ordered sets of clauses $\langle \mathscr{C}, \succeq \rangle$ do not exist. We can approximate these ideal refinement operators in two ways: by dropping the property of properness and by restricting the search space to a finite set. The first approach has been taken by Laird [4] for downward refinement. We have defined a locally finite, complete, and improper upward refinement operator in [3]. The second approach, restricting the search space, has already been taken by Shapiro [15]. However, as we have shown in [1], his downward refinement operator for *reduced* clauses ρ_0 is not complete and hence is not ideal. In the same article we have proposed a new downward refinement operator that is ideal for finite, θ-subsumption ordered sets of reduced clauses. This ideal refinement operator, called ρ_r, will be presented in Section 4.4.

4.1. Introduction

If we want to learn a logical theory that consistently describes an unknown concept (for example, using refinement operators), we do not know the (complexity of the) clauses in the theory to learn. Hence every restriction on a set of clauses that reduces the expressive power brings the risk that target clauses are excluded from this set. If we nevertheless choose to restrict the search space to a finite quasi-ordered set $\langle S, \geq \rangle$, and $C \geq D$ relations are decidable for every pair of elements C, D in S, then ideal refinement operators can be defined in the following way:

$$\rho(C) = dc(C)$$
$$\delta(C) = uc(C).$$

Although a lot of order relation tests are required, algorithms that construct these cover sets can easily be described:

```
$dc(C) := \{D \in S | C > D\}$
while $\exists D, E \in dc(C)$ such that $D \neq E$ and $D \geq E$ do
    $dc(C) := dc(C) \setminus \{E\}$

$uc(C) := \{D \in S | D > C\}$
while $\exists D, E \in uc(C)$ such that $D \neq E$ and $E \geq D$ do
    $uc(C) := uc(C) \setminus \{E\}$.
```

Refinement operators for finite quasi-ordered sets $\langle S, \geq \rangle$ that return cover sets are clearly locally finite and proper. To demonstrate the completeness of this approach, consider an arbitrary pair $C, D \in S$ for which $C > D$. Then either D is a downward cover of C, in which case $E \in \rho(C)$ for some $E \sim D$, or C has a downward cover $E \in \rho(C)$ such that $C > E > D$. In the latter case, $E > D$ holds, and we can extend the ρ-chain from C to D by searching for an $F \in \rho(E)$ such that $F \geq D$. Since $\geq$ is a transitive relation, there are no cycles in the chain from C to D, and since S is finite, this chain is of finite length. Completeness of δ can be shown in a similar way.

The above-defined cover set refinement operators are interesting, mostly for theoretical reasons. They demonstrate the existence of ideal refinement operators in finite quasi-ordered sets, and they demonstrate this in some minimal way, that is, every refinement operator that returns fewer one-step refinements will not be ideal (cf. Lemma 3.1). In practice, the $\geq$ tests that are involved in the computation of cover sets will be too time consuming, particularly when they refer to θ-subsumption tests between clauses. Instead, one-step refinements are preferably determined in a constructive way, i.e., by making (small) modifications to the element that is subjected to refinement. The earlier mentioned minimality also suggests how practical (constructive) ideal refinement operators can be made more efficient. If ρ^+ is an ideal downward refinement operator that returns proper supersets of cover sets, then for every $C \in S$ we can remove from $\rho^+(C)$ the elements $\rho^+(C) \setminus dc(C)$ without losing idealness. However, although the removal of redundant refinements reduces the memory requirements, it also reintroduces $\geq$ tests. Whether this removal is beneficial depends on the ordered search space under attention and should be determined empirically.

4.2. θ-Subsumption and Reduced Clauses

Before we define our ideal refinement operator for finite, θ-subsumption ordered sets of clauses, we discuss Plotkin's [10] notion of reduction and our inverse reduction algorithm. In words, a clause is reduced if it is not equivalent to a proper subset of itself. If a clause D is not reduced, then Plotkin's reduction algorithm finds a reduced clause C that is equivalent to D by removing redundant literals. We call such a C *a reduced equivalent of* D. It is proved by Plotkin that if two equivalent clauses are both reduced, then they are equal, up to renaming variables, i.e., they are alphabetical variants. Since we treat alphabetical variants as the same, all clauses in $\mathscr{R}$ (the set of all reduced clauses in a first-order language $\mathscr{L}$) are nonequivalent.

In a (learning) system that uses a search space of logic formulae, it often is a waste of time and memory to examine more than one clause of an equivalence class. Since for any two clauses C and D, if $C \sim D$, then $C \models E$ iff $D \models E$, using one reduced clause as a representative of every equivalence class might lead to more efficient (learning) systems. However, as will be shown in Section 4.4, θ-subsumption has some unexpected properties that cause problems when we search for clauses in a search space $\mathscr{R}$. These problems will be solved by consulting nonreduced clauses in an intermediate step. We therefore want to build a simple algorithm that reverses the reduction process. We need the following lemma and theorem for this algorithm.

Lemma 4.1. Let C be a clause. If θ is a substitution such that $C\theta = C$, then for some natural number k, $L\theta^k = L$, for all literals L in C.

PROOF. θ must be injective: if $L_1\theta = L_2\theta$ for different $L_1, L_2 \in C$, then θ would decrease the number of literals in C, i.e., $|C\theta| < |C|$, which contradicts $C\theta = C$. For every literal L in C, consider the following sequence:

$$L, L\theta, L\theta^2, L\theta^3, \ldots .$$

Since $C = C\theta = C\theta^2 = \cdots$, and since C is finite, not all $L\theta^i$ can be different. Then for some $i, j, i < j$, we have $L\theta^i = L\theta^j$. Because θ is injective, this implies $L\theta^{j-i} = L$.

For every L, let $n(L)$ be the smallest number such that $L\theta^{n(L)} = L$. Then $L\theta^i = L$ if i is a multiple of $n(L)$. Let k be the least common multiple of all $n(L)$. Then $L\theta^k = L$ for all $L \in C$. □

Lemma 4.2. Let C be a reduced clause, and let D be a clause such that $C \subseteq D$ and $C \sim D$. Then there exists a substitution θ such that $D\theta = C$ and $L\theta = L$ for all literals $L \in C$.

PROOF. Since D θ-subsumes C, $\exists\sigma$: $D\sigma \subseteq C$, which together with $C\sigma \subseteq D\sigma$ implies $C\sigma \subseteq C$. If $D\sigma \subset C$, then also $C\sigma \subset C$, and C would not be reduced. We conclude that $C\sigma = C$. By Lemma 4.1, we then know that for some k, $L\sigma^k = L$ for all $L \in C$, and we define $\theta = \sigma^k$. □

4.3. *Inverse Reduction*

Given a clause D, a reduction algorithm finds an equivalent reduced clause C such that $C \subseteq D$. In this section we develop an algorithm which, given a reduced clause C, constructs supersets D of C that are subsume-equivalent with C. Clearly, none of the proper supersets of C will be reduced.

If we consider a search space $\mathscr{C}$ (all clauses of a first-order language $\mathscr{L}$) then there are infinitely many subsume-equivalent clauses for every nonempty clause C in $\mathscr{C}$. We therefore have to limit the scope of the inverse reduction algorithm before any clause can be processed by it. We accomplish this by bounding the number of literals in D. If C is a reduced clause and m is a fixed positive integer, then we want to find an alphabetical variant of every clause that is equivalent to C and contains less than or exactly m literals.

4.3.1. Inverse Reduction Algorithm. Let C be a reduced clause and let $m \geq 0$ be a fixed natural number. The following algorithm finds an alphabetical variant of every clause that is equivalent to C with m or fewer literals.

```
Let $l = 0$, if $|C| \leq m$, then output $C$
While $l < (m - |C|)$ do
    $l := l + 1$
    For every sequence $L_1, \ldots, L_l$,
    where every $L_i \in C$, but the $L_i$'s are not necessarily distinct.
        Find all sets $E = \{M_1, \ldots, M_l\}$ such that
            $M_i \neq L_i$, $M_i\theta = L_i$ for all $i$, and
            $\theta = \{X_1/t_1, \ldots, X_m/t_m\}$, $X_j \notin var(C)$ for all $j$;
        For every such $E$ output $C \cup E$.
```

Lemma 4.3. Let C be a reduced clause and let $m \geq 0$ be a fixed natural number. Then the inverse reduction algorithm finds an alphabetical variant of every clause that is equivalent to C with m or fewer literals.

PROOF. (Soundness) Let $D = C \cup E$ be a clause that is generated by the inverse reduction algorithm, where $E = \{M_1, \ldots, M_l\}$. Then $D\theta = C$ for some $\theta = \{X_1/t_1, \ldots, X_n/t_n\}$, where no X_j occurs in C. Since $C \subseteq D, C$ θ-subsumes D, and since $D\theta = C$, D θ-subsumes C. Thus C and D are equivalent. Furthermore, since $|D| \leq |C| + |E|$, where $|E| \leq l \leq (m - |C|)$, we have $|D| \leq |C| + m - |C| = m$. That is, D contains m or fewer literals.

(Completeness) Let D' be a clause that is equivalent to C that contains m or fewer literals. We prove that our inverse reduction algorithm finds a clause D that is an alphabetical variant of D'.

Let C' be a reduced clause such that $C' \subseteq D'$ and $C' \sim D'$. C' must be an alphabetical variant of C ([10]). We can rename the variables in D' to find a variant D of D' such that $C \subseteq D$.

Let $E = D \backslash C$. Since C is reduced, $C \subseteq D$ and $C \sim D$, Lemma 4.2 states that there exists a substitution θ such that $D\theta = C$ and $L\theta = L$ for all $L \in C$. This θ does not affect any variable that occurs in C. Furthermore, since $D\theta = C$, for every literal $M_i \in E$ we can find a literal $L_i \in C$ such that $M_i\theta = L_i$.

Summarizing, for every clause D' that is equivalent to C and contains m or less literals, we can define a clause $D = C \cup \{M_1 \ldots, M_l\}$ that is an alphabetical variant of D'. We can construct a sequence of literals $L_1, \ldots, L_l$ in C such that $M_i\theta = L_i$ for all i, where θ does not affect the variables in C. Thus, considering all such sequences $L_1, \ldots, L_l$ and all such substitutions θ, an alphabetical variant of every clause that is equivalent to C with m or fewer literals will be found. □

Example. Let $C = p(X, X) \leftarrow$. Then for $m = 2$, possible redundant literals M_1 are $p(Y, Z), p(X, Y), p(Y, X)$, and $p(Y, Y)$. For $m = 3$, some of the possible M_1's, M_2's, and corresponding θ's are

M_1	M_2	θ
$p(X,Y)$	$p(Y,Z)$	$\{Y/X, Z/X\}$
$p(X,Y)$	$p(X,Z)$	$\{Y/X, Z/X\}$
$p(X,Y)$	$p(Z,W)$	$\{Y/X, Z/X, W/X\}$
$p(X,Y)$	$p(Y,X)$	$\{Y/X\}$
$p(Y,Y)$	$p(Z,Z)$	$\{Y/X, Z/X\}$

REMARK. Note that the algorithm above does not find *all* clauses D with $|D| \leq m$ that are subsume-equivalent to C. For example, given the reduced clause $C = p(X) \leftarrow q(X, X)$, it will not find the subsume-equivalent clause $D' = p(U) \leftarrow q(U, U), q(U, V)$. However, for every such clause D', an alphabetic variant D of D' will be found such that $D = C \cup E$. For the just defined clause D', this could be $D = p(X) \leftarrow q(X, X), q(X, Y)$.

The inverse reduction algorithm above is theoretically interesting because it shows the structure of all clauses in the equivalence class of a (reduced) clause. It can also be used more practically. For example, in Section 4.4 we define our ideal downward refinement operator ρ_r, in which inverse reduction plays an important role for completeness.

4.4. A New Refinement Operator for Reduced Clauses

When we define an equivalence relation on a set, it is usually required that the equivalence relation is compatible with the important operations on this set, i.e., operations on different members of the same equivalence class yield equivalent results. Unfortunately, this is not true for the equivalence relation that is induced by θ-subsumption.

Example. Consider the following clauses:

$C = q \leftarrow p(X,Y)$

$D = q \leftarrow p(X,Y), p(X,Z).$

$C \subset D$ implies $C \succeq D$, and, by $\theta = \{Z/Y\}$, we get $D\theta = C$, which implies $D \succeq C$. Thus $C \sim D$, i.e., C and D are subsume-equivalent. It can be verified that C is a reduced clause. D is clearly not reduced because it is equivalent to its proper subset C; the literal $p(X,Y)$ in the body of D is redundant. If we define $\sigma = \{X/Z\}$, then we get

$C\sigma = q \leftarrow p(Z,Y)$

$D\sigma = q \leftarrow p(Z,Y), p(Z,Z).$

$C\sigma$ is reduced but $D\sigma$ is not, and it can be reduced to $q \leftarrow p(Z,Z)$. Thus, whereas C and D were subsume-equivalent, $C\sigma$ and $D\sigma$ are not, because their reduced equivalents are not alphabetical variants. A similar situation occurs when we add the literal $L = \neg\, p(Y,Z)$ to C and D:

$C \cup \{L\} = q \leftarrow p(X,Y), p(Y,Z)$

$D \cup \{L\} = q \leftarrow p(X,Y), p(X,Z), p(Y,Z).$

Now, both $C \cup \{L\}$ and $D \cup \{L\}$ are reduced and they are not alphabetical variants. Thus, whereas C and D were subsume-equivalent, $C \cup \{L\}$ and $D \cup \{L\}$ are not.

The example above illustrates that subsume-equivalent clauses may be non-equivalent after application of a substitution or after addition of a literal. Therefore, if literal additions and substitutions are some of the operations on clauses, it might be insufficient to consider only reduced representatives of equivalence classes. In fact, Shapiro [15] has defined a downward refinement operator for reduced clauses only, ρ_0, in which every refinement operation is either a substitution or the addition of a literal. As is shown in [1] and [7], this refinement operator is not complete. The main causes of this incompleteness are the inability to add more than one literal in one refinement step, and a size restriction that prohibits "decreasing substitutions" that result in refinements that contain fewer literals than the refined clause. For a detailed discussion on the incompleteness of ρ_0, we refer to [1].

To overcome the completeness problems of Shapiro's ρ_0, we drop his "nondecreasing" restriction on refinements, thus allowing refinements to have fewer literals than the refined clause. For example, $q \leftarrow p(X,X)$ is a one-step refinement of $q \leftarrow p(X,Y), p(Y,X)$ that is obtained by unifying the variables X and Y. We also introduce intermediate nonreduced clauses through which we can add more than one literal in one refinement step. The purpose of these nonreduced clauses is to form a bridge between the reduced refined clause and reduced proper specializations that are not derivable otherwise. The following example illustrates this idea.

Example. Consider a search space of reduced clauses that contains, among others, the following clauses:

$C = q \leftarrow p(X,Y), p(Y,Z), p(Z,X)$

$D = q \leftarrow p(X,Y), p(Y,Z), p(Z,X), p(X,W), p(W,X).$

Then C and D are both reduced and, as can be verified, $C \succ D$. So any complete refinement operator ρ satisfies $D \in \rho^*(C)$. If we were to add one of the literals in $D \backslash C$ to C, then we obtain a nonreduced clause that is equivalent to C. Thus, to derive D from C, we must somehow be able to add more than one literal in one refinement step. In our approach this is achieved by first computing a nonreduced intermediate clause that is equivalent to C, for example,

$$C' = q \leftarrow p(X,Y), p(Y,Z), p(Z,X), p(X,V), p(W,X).$$

From this intermediate clause C' we can derive D by unifying the variables V and W.

In Section 4.3 we have presented our inverse reduction algorithm. Given a reduced clause C and an integer m, this algorithm generates clauses C' with m or fewer literals that are equivalent to C. We will use $eq(C)$ to denote the set of all such C''s. Every clause $C' \neq C$ in this set satisfies $C' = C \cup E$ for some set of literals E. Since $|E|$ can be larger than 1, we can use inverse reduction to add more than one literal in one refinement step. By applying substitutions to these intermediate clauses C', we solve the problem presented in the example above.

Note that for every redundant literal in E that is added to C, we can find a literal in C with the same predicate name and sign.

4.4.1. Compatible Literal. Two literals L and M are *compatible* if they have the same predicate symbol and sign.

If we want to add literals that are incompatible with every literal in a reduced clause C, adding one literal in one refinement step is sufficient.

4.4.2. Most General Literal. A literal L is called *most general with respect to* a clause C if L has only distinct variables as arguments that do not occur in C.

Lemma 4.4. Let C be a clause, and let L be a most general literal with respect to C. Then the following two conditions are equivalent:

1. *C properly θ-subsumes $D = C \cup \{L\}$.*
2. *For any literal M in C, L and M are incompatible.*

PROOF. $1 \Rightarrow 2$: Assume that 2 does not hold. Then there is an M in C such that M and L have the same predicate name and sign. Let θ be defined on variables of L only, such that $L\theta = M$. Then $D\theta = (C \cup \{L\})\theta = C$. This means that D also θ-subsumes C. Therefore, $C \sim D$.

$2 \Rightarrow 1$: $C \subset D$, so clearly C θ-subsumes D. Assume that also D θ-subsumes C, then for some θ, $D\theta \subseteq C$. But then also $L\theta \in C$, and $L\theta$ and L must have the same predicate name and sign. □

It is easy to verify that, if C is reduced and L satisfies the conditions of Lemma 4.4, then $D = C \cup \{L\}$ is also reduced.

Apart from the way in which we restrict the search space, we have introduced all notions that are relevant for the definition of our refinement operator ρ_r. As was mentioned earlier, this restriction is necessary because in an unrestricted search space $\mathscr{C}$, for every nonempty clause, infinitely many equivalent but unequal clauses can be constructed. In Section 4.5, we will define a complexity measure *newsize* that can be used to bound unrestricted sets of clauses ($\mathscr{C}$ or $\mathscr{R}$) to finite subsets ($\mathscr{C}^{newsize}$ or $\mathscr{R}^{newsize}$). Although this restriction has some impact on the definition and completeness of ρ_r, we will not discuss these matters until that section. For the time being we just state that $\mathscr{C}^{newsize}$ and $\mathscr{R}^{newsize}$ are finite sets of (reduced) clauses.

Definition of ρ_r. Let C be a reduced clause. Then $D \in \rho_r(C)$ iff D is reduced and one of the following conditions holds:

1. $C \succ D$, and there are $C' \in eq(C)$ and $D' \in eq(D)$ such that $D' = C'\theta$, where $\theta = \{X, Y\}$ and both X and Y occur in C'.
2. $C \succ D$, and there are $C' \in eq(C)$ and $D' \in eq(D)$ such that $D' = C'\theta$, where $\theta = \{X/f(Y_1, \dots, Y_n)\}$, f is an n-place function symbol, X occurs in C', and all Y_i's are distinct variables not in C'.
3. $D = C \cup \{L\}$, where L is a most general literal with respect to C that is incompatible with every literal in C.

REMARK. Our main interest in the just defined refinement operator ρ_r are theoretical and concern its ideal properties. As we will prove in the remainder of this section, it is ideal for finite, reduced search spaces bounded by *newsize*.

Although ρ_r can be implemented, it is not very practical for several reasons. First, computing refinements is a rather complex operation. For example, the number of clauses generated by the inverse reduction algorithm, which is used to compute C' given C, grows exponentially with the size of the search space. Second, the refinement operator is not fully constructive. We can reformulate the first item as follows, to make it more constructive:

1. $C' \in eq(C)$, $D' = C'\theta$, where $\theta = \{X/Y\}$ and both X and Y occur in C', and D is the reduced equivalent of D' and $C \succ D$.

Still, expensive tests ($C \succ D$) are included to guarantee properness. We do not know whether these tests can be avoided without losing properness.

Theorem 4.1. Let $C, D \in \mathscr{R}$ be reduced clauses such that $C \succ D$. Then there is a ρ_r-chain from C to D.

Lemma 4.5. Let $C, D \in \mathscr{R}$ be reduced clauses such that $C \succ D$ and let $C' \in eq(C)$, $D' \in eq(D)$ satisfy $C'\theta = D'$. Then there is an $E \in \rho_r(C)$ such that $E \succeq D$.

PROOF. Let $C' = C_0, \dots, C_n = D'$ be a chain of clauses such that $C_i = C_{i-1}\theta_i$, $1 \le i \le n$, where every θ_i is a substitution as defined in ρ_r's item 1 or 2. Reynolds [13, proof of Theorem 4] has shown how such a chain of substitutions can be constructed for atoms. The same procedure can be used for clauses. Let C_k be the first C_i that is not equivalent to C. Since $C \succ D$, such a C_k exists. If we let E be the reduced equivalent of C_k, then $C_{k-1} \in eq(C)$, $C_k = C_{k-1}\theta_k$, and $E \in \rho_r(C)$. Furthermore, $C_k \succeq D'$ (by $C_k\theta_{k+1} \cdots \theta_n = C_n = D'$), which together with $E \sim C_k$ and $D' \sim D$ implies $E \succeq D$. □

The following example illustrates the proof of Lemma 4.5.

Example. Consider the following clauses:

$$C = p \leftarrow q(a,W), q(X,b), q(c,Y), q(Z,d)$$

$$D = p \leftarrow q(a,b), q(c,b), q(c,d), q(a,d).$$

Let $C' = C$ and let $D' = D$; then, by $\theta = \{W/b, X/c, Y/d, Z/a\}$, $C'\theta = D'$. θ can be split into the ρ_r-substitutions $\theta_1 = \{W/b\}$, $\theta_2 = \{X/c)\}$, $\theta_3 = \{Y/d\}$, and $\theta_4 = \{Z/a\}$. We then get the following chain of clauses:

$$C_0 = C = p \leftarrow q(a,W), q(X,b), q(c,Y), q(Z,d)$$
$$C_1 = C_0\theta_1 = p \leftarrow q(a,b), q(X,b), q(c,Y), q(Z,d)$$
$$C_2 = C_1\theta_2 = p \leftarrow q(a,b), q(c,b), q(c,Y), q(Z,d)$$
$$C_3 = C_2\theta_3 = p \leftarrow q(a,b), q(c,b), q(c,d), q(Z,d)$$
$$C_4 = C_3\theta_4 = p \leftarrow q(a,b), q(c,b), q(c,d), q(a,d).$$

C_1 is the first clause that is not equivalent to C. The reduced equivalent E of C_1 is

$$E = p \leftarrow q(a,b), q(c,Y), q(Z,d),$$

and E is a member of $\rho_r(C)$ that θ-subsumes D.

Lemma 4.6. Let $C, D \in \mathscr{R}$ be reduced clauses such that $C \succ D$ and $C \subset D$. Then there is a $E \in \rho_r(C)$ such that $E \succeq D$.

PROOF. Let F be a maximum subset of $D \setminus C$ such that $(C \cup F) \sim C$. This means, that for every literal M in $D \setminus (C \cup F)$, $C \succ ((C \cup F \cup \{M\})$. Let L be a most general literal with respect to $C \cup F$ such that $L\theta = M$ for one of those literals.

If $C \cup F \cup \{L\}$ is not equivalent to $C \cup F$, then, by Lemma 4.4, L is incompatible with every literal in $C \cup F$. Thus L is incompatible with every literal in C. Hence, $E = (C \cup \{L\}) \in \rho_r(C)$ and, since $E\theta \subset D$, $E \succeq D$.

Otherwise, $C' = C \cup F \cup \{L\}$ and $D' = C \cup F \cup \{M\}$ satisfy $C' \succ D'$ and $C'\theta = D'$. Using Lemma 4.5 a clause, E can be found such that $E \in \rho_r(C)$ and $E \succeq D'$. Since $D' \subseteq D$, this clause E also satisfies $E \succeq D$. □

The following examples illustrate the proof of Lemma 4.6.

Example. Consider the following clauses:

$$C = p(X) \leftarrow$$
$$D = p(X) \leftarrow q(a,X).$$

The only subset F of $D \setminus C$ such that $(C \cup F) \sim C$ is $\{\ \}$, the empty set. $M = \neg q(a, X)$ is the only literal in $D \setminus (C \cup \{\ \})$. $L = \neg q(Y, Z)$ is most general w.r.t. C and $L\theta = M$ for $\theta = \{Y/a, Z/X\}$. $C \cup \{L\}$ is reduced and

$$E = p(X) \leftarrow q(Y,Z)$$

is a member of $\rho_r(C)$ that θ-subsumes D.

Example. Consider the following clauses:

$$C = p(X) \leftarrow q(X,a)$$
$$D = p(X) \leftarrow q(X,a), q(Y,Z), q(Z,Y).$$

$F = \{\neg q(Y,Z)\}$ is a maximum subset of $D \setminus C$ such that $(C \cup F) \sim C$. Taking $M = \neg q(Z,Y)$, we get $L = \neg q(U,V)$ as a most general literal with respect to $C \cup F$. $C' = C \cup F \cup \{L\}$ is equivalent to C, and C' properly θ-subsumes $D' = C \cup F \cup \{M\} (= D)$:

$$C' = p(X) \leftarrow q(X,a), q(Y,Z), q(U,V)$$
$$D' = p(X) \leftarrow q(X,a), q(Y,Z), q(Z,Y).$$

By Lemma 4.5 we can find a refinement E of C that θ-subsumes D'. In the proof of Lemma 4.5, we get $C'\theta = D'$ for $\theta = \{U/Z, V/Y\}$. This substitution θ can be split into $\theta_1 = \{U/Z\}$ and $\theta_2 = \{V/Y\}$. We now get the following chain of clauses:

$$C_0 = C'$$

$$C_1 = C_0\theta_1 = p(X) \leftarrow q(X,a), q(Y,Z), q(Z,V)$$

$$C_2 = C_1\theta_2 = p(X) \leftarrow q(X,a), q(Y,Z), q(Z,Y)$$

C_1 is the first clause that is not equivalent to C'. The reduced equivalent E of C_1 is

$$E = p(X) \leftarrow q(X,a), q(Y,Z), q(Z,V),$$

and E is a member of $\rho_r(C)$ that θ-subsumes D.

PROOF OF THEOREM 4.1. For every pair of reduced clauses C and D such that $C \succ D$, we can find a substitution θ for which $C\theta \subseteq D$. Let F be the reduced equivalent of $C\theta$; then either $C \succ F$ or $C \sim F$.

If $C \succ F$, then C and F satisfy the conditions of Lemma 4.5. Otherwise, $F \in eq(C)$, and F and D satisfy the conditions of Lemma 4.6.

In both cases the first element E of a ρ_r-chain from C to D can be found. We can complete a ρ_r-chain from C to D by repeatedly finding the first element in a chain from E to D. In Lemma 4.7 in the next subsection, we prove that this chain is of finite length. □

4.5. A New Complexity Measure

Shapiro had to restrict the search space for ρ_0 to a finite set, because without this restriction there are infinitely many ways to add a literal to a clause in one refinement step [15], and ρ_0 would not be locally finite. In our refinement operator ρ_r, only a finite number of most general literals can be added through ρ_r-item 3. However, as was stated before, for every nonempty clause C, we can construct infinitely many subsume-equivalent clauses. Thus, in an unrestricted search space, the inverse reduction function eq that is used in the definition of ρ_r is not computable. Therefore, we still have to restrict the search space.

In concrete examples of refinement operators, Shapiro restricts the search space using Reynolds' [13] complexity that was originally defined for atoms:

- *rsize*(C) = the number of symbols occurrences in C − the number of distinct variables in C.

Example. Consider the clauses

$$C = p(X) \leftarrow q(X,Y), q(Y,X)$$

$$D_1 = p(X) \leftarrow q(X,X)$$

$$D_2 = p(X) \leftarrow q(X,a), q(a,X).$$

Then C θ-subsumes both D_1 and D_2, where *rsize*(C) = 6, *rsize*(D_1) = 4, and *rsize*(D_2) = 7. Apparently, *rsize* can increase or decrease when a clause is specialized.

We could follow Shapiro and use *rsize* to restrict search space $\mathscr{R}$ to $\mathscr{R}^{rsize}$. But because of the incompatibility of θ-subsumption and *rsize*, we introduce a new complexity measure that is more naturally related to θ-subsumption. Moreover, given C and D, with the property that $C \succ D$ in a search space bounded by our size measure, we can always find a ρ_r-chain from C to some D' in the search space such that $D' \sim D$. We define:

- $newsize(C) = (maxsize(C), |C|)$, where $maxsize(C) = max\{rsize(L) | L \in C\}$.

The following properties of *maxsize* and *newsize* are stated without proof but can easily be verified. The first proposition follows directly from the observation that the number of literals of $rsize \leq k$ is finite.

Proposition 4.1. For every fixed pair of numbers (k, m), the number of clauses with $newsize(C) \leq (k, m)$, i.e., $maxsize(C) \leq k$ and $|C| \leq m$, is finite.

Proposition 4.2. If $C \succeq D$, then $maxsize(C) \leq maxsize(D)$.

From here on, we assume that our refinement operator ρ_r is applied on θ-subsumption ordered sets $\langle \mathscr{R}^{newsize}, \succeq \rangle$. Thus, for some fixed k and m, every clause in $\mathscr{R}^{newsize}$ satisfies $newsize(C) \leq (k, m)$, i.e., $maxsize(C) \leq k$ and $|C| \leq m$. We should add this as a condition to the definition of ρ_r: every one of the clauses C, D, C', D' must satisfy $newsize \leq (k, m)$). Thus the clauses that are to be generated by the inverse reduction algorithm for $eq(C)$ are also restricted by *newsize*.

From Proposition 4.2 it follows that $C \sim D$ implies $maxsize(C) = maxsize(D)$. This relation has a nice consequence for the restricted computation of $eq(C)$. Let C be a clause in $\mathscr{R}^{newsize}$ for some fixed bound (k, m). Then $eq(C)$ should compute all clauses D in $\mathscr{R}^{newsize}$ that are subsume-equivalent to C. Since $D \sim C$ implies $maxsize(D) = maxsize(C)$, $maxsize(D) \leq k$ automatically holds, and we only have to ensure that $|D|$ does not exceed m. Since the maximum number of literals m is a parameter of our inverse reduction algorithm, we can use this algorithm unmodified. Note that if we were to use *rsize* to restrict $\mathscr{R}$, auxiliary tests for $rsize(D) \leq k$ would be required.

By definition of items 1 and 2 and by Lemma 4.4, ρ_r is a proper refinement operator. Local finiteness of ρ_r is guaranteed if every clause that is involved in the computation of $\rho(C)$ satisfies $newsize \leq (k, m)$. Lemmas 4.5 and 4.6 state that if $C \succ D$ and $C\theta = D$ or $C \subset D$, then we can find a successor E of C in a ρ_r-chain from C to D. It can be verified that, if C properly θ-subsumes D and both C and D are members of $\mathscr{R}^{newsize}$, then every clause that is used in the computation of E in these lemmas is in $\mathscr{R}^{newsize}$ as well. To prove that ρ_r is an ideal downward refinement operator for $\langle \mathscr{R}^{newsize}, \succeq \rangle$, it remains to show that ρ_r is complete for such ordered sets. According to the proof of Theorem 4.1, it remains to prove that all ρ_r chains in $\mathscr{R}^{newsize}$ are of finite length.

Lemma 4.7. Let (k, m) be a fixed pair of numbers and let $C_0, C_1, C_2 \cdots$ be a ρ_r chain, where $newsize(C_i) \leq (k, m)$ for every C_i. Then this ρ_r chain is of finite length.

PROOF. For every pair of fixed numbers (k, m) there are finitely many clauses C such that $newsize(C) \leq (k, m)$. So every ρ_r chain contains finitely many different clauses. Since all refinement steps are proper ($D \in \rho_r(C)$ implies $C \succ D$), no clause can occur more than once in a ρ_r chain, and every ρ_r chain is finite. □

Corollary 4.1. *ρ_r is ideal for $\langle \mathscr{R}^{newsize}, \succeq \rangle$.*

REMARK. ρ_r can easily be generalized to a refinement downward refinement operator ρ_c that is ideal for $\langle \mathscr{C}^{newsize}, \succeq \rangle$. For this purpose we only have to reduce the nonreduced clauses that are subjected to refinement.

Definition of ρ_c. Let C be a clause of $\mathscr{C}^{newsize}$, Then $D \in \rho_c(C)$ iff

$$D \in \rho_r(E), \text{ where } E \text{ is a reduced equivalent of } C, \text{ and } D \in \mathscr{C}^{newsize}.$$

Since $newsize(E) \leq newsize(C)$ ($maxsize(E) \leq maxsize(C)$ and $|E| \leq |C|$), E is in the restricted search space whenever C is. Note that all refinements that are computed by ρ_c are reduced clauses. Hence, only nonreduced clauses that have an origin other than ρ_c (for example, clauses that are provided by the user or that are computed by another operator) need to be reduced in the first phase of ρ_c.

For an ideal *upward* refinement operator for $\langle \mathscr{R}^{newsize}, \succeq \rangle$ that is defined very similarly to ρ_r, we refer to our article [9].

5. CONCLUSIONS

In this article we have characterized sufficient conditions in terms of cover sets to conclude when ideal refinement operators for a quasi-ordered set $\langle S, \geq \rangle$ do not exist. We have translated these conditions to uncovered chains and showed that ideal upward and downward refinement operators do not exist for unrestricted sets of clauses that are ordered by θ-subsumption, $\langle \mathscr{C}, \succeq \rangle$. We have defined a new complexity measure for clauses *newsize* to restrict a search space of (reduced) clauses $\mathscr{C}(\mathscr{R})$ to a finite set $\mathscr{C}^{newsize}$ ($\mathscr{R}^{newsize}$) and a downward refinement operator, ρ_r, that is ideal (i.e., locally finite, complete, and proper) for thus restricted ordered sets of reduced clauses $\langle \mathscr{R}^{newsize}, \succeq \rangle$.

REFERENCES

1. van der Laag, P. R. J. and Nienhuys-Cheng, S. H., Subsumption and Refinement in Model Inference, in: P. B. Brazdil (ed.), *Proc. European Conf. Machine Learning (ECML-93), Lecture Notes in Artificial Intelligence*, Springer-Verlag, Berlin, 1993, Vol. 667, pp. 95–114.
2. van der Laag, P. R. J. and Nienhuys-Cheng, S. H., A Note on Ideal Refinement Operators in Inductive Logic Programming, in: S. Wrobel (ed.), *Proc. ILP-94*, Technical Report GMD-Studien no. 237, GMD, Germany, 1994, pp. 247–260.
3. van der Laag, P. R. J. and Nienhuys-Cheng, S. H., Existence and Nonexistence of Complete Refinement Operators, in: F. Bergadano and L. De Raedt (eds.), *Proc. European Conf. Machine Learning (ECML-94), Lecture Notes in Artificial Intelligence*, Springer-Verlag, Berlin, 1994, Vol. 784, pp. 307–322.
4. Laird, P. D., *Learning from Good and Bad Data*, Kluwer Academic Publishers, Dordrecht, The Netherlands, 1988.
5. Ling, C. and Dawes, M., SIM the Inverse of Shapiro's MIS, Technical Report 263, Department of Computer Science, University of Western Ontario, London, ON, Canada, 1990.
6. Muggleton, S. and De Raedt, L., Inductive Logic Programming: Theory and Methods, *J. Logic Programming* 12:629–679 (1994).

7. Niblett, T., A Note on Refinement Operators, in: P. B. Brazdil (ed.), *Proc. European Conf. Machine Learning (ECML-93), Lecture Notes in Artificial Intelligence*, Springer-Verlag, Berlin, 1993, Vol. 667, pp. 329–335.
8. Nienhuys-Cheng, S. H. and de Wolf, R., *Foundations of Inductive Logic Programming*, Springer-Verlag, 1997, to appear.
9. Nienhuys-Cheng, S. H., van der Laag, P. R. J., and van der Torre, L. W. N., Constructing Refinement Operators by Decomposing Logical Implication, in: P. Torasso (ed.), *Proc. Third Congress Italian Association Artif. Intell. (AI*IA '93), Lecture Notes in Artificial Intelligence*, Springer-Verlag, Berlin, 1993, Vol. 728, pp. 178–189.
10. Plotkin, G. D., A Note on Inductive Generalization, *Machine Intell.* 5:153–163 (1970).
11. Quinlan, J. R., Learning Logical Definitions from Relations, *Machine Learning* 5: 239–266 (1990).
12. De Raedt, L. and Bruynooghe, M., A Theory of Clausal Discovery, in: S. H. Muggleton (ed.), *Proc. ILP-93*, Technical Report IJS-DP-6707, J. Stefan Institute, Slovenia, 1993, pp. 25–40.
13. Reynolds, J. C., Transformational Systems and the Algebraic Structure of Atomic Formulas, *Machine Intell.* 5:135–153 (1970).
14. Semeraro, G., Brunk, C. A., and Pazzani, M. J., Traps and Pitfalls When Learning Logical Theories: A Case Study with FOIL and FOCL, Technical Report 93-33, University of California, Irvine, CA, July 1993.
15. Shapiro, E. Y., Inductive Inference of Theories from Facts, Technical Report 192, Department of Computer Science, Yale University, New Haven, CT, 1981.
16. Van Laer, W., Inductief Afleiden van Logische Regels, Master's Thesis, Department of Computing Science, Katholieke Universiteit Leuven, Leuven, Belgium, 1993 (in Dutch).