# Existence and Nonexistence of Complete Refinement Operators

Patrick R. J. van der Laag[1,2] and Shan-Hwei Nienhuys-Cheng[1]

[1] Department of Computer Science, Erasmus University of Rotterdam, P.O.Box 1738, 3000 DR Rotterdam, the Netherlands
[2] Tinbergen Institute

**Abstract.** Inductive Logic Programming is a subfield of Machine Learning concerned with the induction of logic programs. In Shapiro's Model Inference System – a system that infers theories from examples – the use of downward *refinement operators* was introduced to walk through an ordered search space of clauses. Downward and upward refinement operators compute specializations and generalizations of clauses respectively. In this article we present the results of our study of *completeness and properness* of refinement operators for an *unrestricted search space* of clauses ordered by *$\theta$-subsumption*. We prove that locally finite downward and upward refinement operators that are both complete and proper for unrestricted search spaces ordered by $\theta$-subsumption do not exist. We also present a complete but improper upward refinement operator. This operator forms a counterpart to Laird's downward refinement operator with the same properties.

## 1 Introduction

Inductive Logic Programming (ILP) is a subfield of machine learning concerned with the induction of logic programs that are consistent with examples of an unknown concept, i.e. programs that can derive all positive examples and none of the negative ones. Within ILP, generalization and specialization of theories and clauses play important roles.

Whereas logical implication between theories or clauses are conceptually most desirable, $\theta$-subsumption, a weaker version of it, is widely used since it is more manageable and we use it as our notion of generality w.r.t. which we study refinement. Clause $C$ *$\theta$-subsumes* clause $D$ if there is a substitution $\theta$ such that $C\theta \subseteq D$, where $C$ and $D$ are represented as sets of literals. If $C$ $\theta$-subsumes $D$ and $D$ $\theta$-subsumes $C$ then $C$ and $D$ are called *(subsume) equivalent.*

Shapiro [15] has introduced the use of downward refinement operators in model inference. A *downward refinement operator* $\rho$ can be used to derive a set of specializations of a clause $C$, denoted by $\rho(C)$. We also consider *upward refinement operators*, denoted by $\delta$, that return sets of generalizations. In our opinion, ideal refinement operators are *locally finite*, *complete* and, less important, *proper* for search spaces of clauses that are not restricted beforehand. Local finiteness means that $\rho(C)$ or $\delta(C)$ is finite and computable. Properness means that $C$ is

not equivalent with any element of $\rho(C)$ or $\delta(C)$, and completeness that every proper specialization or generalization of a clause can be found.

Although $\theta$-subsumption is simpler and easier to understand than logical implication, there are still some simple looking but unanswered questions related to refinement w.r.t. $\theta$-subsumption. They will be answered in this article.

1. Can we define locally finite, complete and proper downward or upward refinement operators for unrestricted search spaces ordered by $\theta$-subsumption?
2. If not, can we define locally finite and complete downward or upward refinement operators for such search spaces if we drop the condition of properness?

The following example illustrates the problems of completeness and properness of refinement operators for unrestricted search spaces.

*Example 1.* Consider the following clauses that represent that node $X$ is in a cycle of length 3 and 1 respectively:

$C = cycle(X) \leftarrow con(X,Y), con(Y,Z), con(Z,X)$
$D = cycle(X) \leftarrow con(X,X)$

Then clause $C$ $\theta$-subsumes clause $D$ as can be verified by the substitution $\{Y/X, Z/X\}$. A downward refinement operator like Laird's $\rho_0$ [7] can derive $D$ from $C$ in two refinement steps: applying the variable unifications $\{Y/Z\}$ and $\{Z/X\}$. We say that there is a $\rho_0$-*chain* from $C$ to $D$ via $E$.

$E = cycle(X) \leftarrow con(X,X), con(X,Z), con(Z,X) \in \rho_0(C)$
$D' = cycle(X) \leftarrow con(X,X), con(X,X), con(X,X) \in \rho_0(E) \subseteq \rho_0^2(C)$

In ILP, clauses are usually interpreted as sets of literals, hence the duplicate literals in $D'$ can be removed to get $D$. When we want to derive $C$ from $D$ using an upward refinement operator, anti-unification of variables in $D$ can only result in $C$ if we first duplicate the literal $con(X,X)$ in $D$ twice. Note that the clauses $D$ and $E$ are subsume equivalent. Still it seems useful to derive $E$ from $D$ first, since it can be used to derive $C$ later on. We will prove that this kind of equivalent refinement steps are necessary for completeness.

In this example, two literal duplications were sufficient to derive $C$. But how many duplications are necessary to derive other clauses in an unrestricted search space, for example cycles of arbitrary length $n$? Even if we drop the condition of properness, problems with local finiteness of a complete upward refinement operator arise if the required number of literal duplications cannot be determined.

The results in this article provide a negative answer to Question 1. We prove that complete and proper refinement operators for unrestricted search spaces ordered by $\theta$-subsumption do not exist.

Question 2 is already partly answered by Laird's [7] improper, complete downward refinement operator for unrestricted sets of clauses. In this article we complete the affirmative answer by defining an upward counterpart to Laird's downward refinement operator. We will show that the number of required literal

duplications in upward refinement is finite and computable. Using this observation we will come to our positive result,

In practice these results imply that under the restriction of local finiteness and completeness, any attempt to modify an improper refinement operator into a proper one or to construct a new proper refinement operator is doomed to fail.

**Related work.** Refinement operators for clauses ordered by $\theta$-subsumption are also described by Shapiro [15], Laird [7], and Ling and Dawes [9]. Shapiro intended to define a downward refinement operator for finite search spaces such that every reduced clause was derivable from the empty clause. We have shown that his operator did not satisfy this weak completeness property[3] and proposed another proper and complete downward refinement operator for finite search spaces of reduced clauses [6]. Laird's modified version of Shapiro's downward refinement operator is complete but improper for unrestricted search spaces as will be discussed in Section 4. Ling and Dawes have proposed an upward refinement operator for clauses that is, using our definitions, neither complete nor proper and operates on finite search spaces. It lacks the, in our opinion vital (cf. Example 1), ability of increasing the number of literals in generalization. In [12], the authors and Leon van der Torre presented a deconstruction of logical implication that resulted in six downward and upward refinement operators for finite search spaces ordered by six increasingly strong orderings. In this last article the use of the substitution and set ordering to define refinement operators for the $\theta$-subsumption ordering was introduced. This approach will also be taken in Section 5, in which we develop our complete upward refinement operator for unrestricted search spaces. This last refinement operator is presented before as a working paper in [5].

The difference between $\theta$-subsumption and logical implication between clauses can be characterized by self-resolution. When a clause is resolved $n-1$ times with itself then the resulting clause is called an $n$-th power of the original clause. The original clause is also called an $n$-th root of the resulting clause [10]. Operators that compute $n$-th powers and roots of a clause can be used to extend downward and upward refinement operators for $\theta$-subsumption to logical implication. For example, the complete upward refinement operator for $\theta$-subsumption that will be presented in this article becomes complete for logical implication when we incorporate Idestam-Almquist's [1] *expansions* of clauses. Incomplete but more efficient operators that compute $n$-th roots of clauses are described in [2] and [8].

**Outline of the article.** In Section 2 we give some basic definitions concerning orderings and refinement operators. In Section 3 we prove the nonexistence of complete and proper refinement operators for unrestricted search spaces ordered by $\theta$-subsumption. In Section 4 we briefly discuss Laird's complete downward refinement operator for such search spaces. A complete upward refinement operator will be defined in three steps in Section 5. Finally, in Section 6, we will present our conclusions and suggest some future research directions.

[3] This incompleteness is described independently by Niblett [11]

# 2 Notation and Definitions

## 2.1 Notation

Given a language of first order logic $\mathcal{L}$ with finitely many function and predicate symbols we use the following notation. Clauses are denoted by $C, D, \ldots$, function symbols by $f, g$, constants by $a$, $b$, predicate symbols by $p, q, r$ and literals by $L, M$. All these symbols can occur with subscripts.

In this article we make an explicit distinction between the representation of a clause as a set and as a sequence of literals. This is necessary since we need to describe the operation of duplicating a literal in Section 5. In the four preceding sections the difference is not important. Whenever we say 'clause $C$' we mean a *sequence* of literals: $C = L_1, \ldots, L_m \leftarrow M_1, \ldots, M_n$.

The *set* representation is common in ILP and will, in this article, sometimes be used to facilitate definitions. By writing $\dot{C}$ we mean that clause $C$ is considered as a set of literals and thus the internal ordering and repetition of literals play no role. For example, the clauses

$C = even(X) \leftarrow odd(Y), odd(Y), plus(Y, Y, X)$, and
$D = even(X) \leftarrow plus(Y, Y, X), odd(Y)$

have the same set representation

$$\dot{C} = \dot{D} = \{even(X), \neg plus(Y, Y, X), \neg odd(Y)\}.$$

All definitions and properties of refinement operators in this article will be described in terms of general first order clauses but they can easily be adapted for (definite) Horn-clauses.

## 2.2 Definitions

In the following definitions $S$ can be any set of clauses and $\succeq$ can be any ordering. Given two literals $L$ and $M$, we use the following notions:

- $L$ and $M$ are called *compatible* iff they have the same predicate name and sign [13].
- A literal is called *most general w.r.t. a clause* $C$ iff it contains only distinct variables as arguments that do not occur in $C$ [15].

Given a set of clauses $S$ and clauses $C, D, E \in S$, we use the following notions:

- $S$ is called *unrestricted* iff all clauses of some language $\mathcal{L}$ are in it.
- A binary relation $\succeq$ on $S$ is called a *quasi-ordering* on $S$ iff it is reflexive ($C \succeq C$) and transitive ($C \succeq D$ and $D \succeq E$ imply $C \succeq E$). For every quasi-ordering $\succeq$ we can define an *equivalence* relation $\sim$ by $C \sim D$ iff $C \succeq D$ and $D \succeq C$.
- Quasi-ordering $\succeq_1$ is *stronger* than $\succeq_2$ if $C \succeq_2 D$ implies $C \succeq_1 D$. If also for some $C, D$, $C \not\succeq_2 D$ and $C \succeq_1 D$ then $\succeq_1$ is *strictly stronger* than $\succeq_2$.

- If $C \succeq D$ holds then $C$ is called a *generalization* of $D$ and $D$ a *specialization* of $C$. If $C \succ D$ holds, meaning $C \succeq D$ and $D \not\succeq C$, than $C$ is a *proper generalization* of $D$ and $D$ is *proper specialization* of $C$. If $C \succeq D$ or $D \succeq C$ then $C$ and $D$ are called *comparable*.
- If $C \succ D$ and there exists no $E$ such that $C \succ E \succ D$, then $C$ is called an *upward cover* of $D$ and $D$ is called a *downward cover* of $C$.

Given a set of clauses $S$ ordered by $\succeq$,

- $\rho$ is a *downward refinement operator* iff $\forall C \in S$: $\rho(C) \subseteq \{D \in S | C \succeq D\}$
- $\delta$ is an *upward refinement operator* iff $\forall C \in S$: $\delta(C) \subseteq \{D \in S | D \succeq C\}$

All definitions regarding refinement operators will be presented in terms of downward refinement but are defined similarly for upward refinement.

- $\rho$ is called *locally finite* iff $\forall C \in S$: $\rho(C)$ is finite and computable.
- $\rho$ is called *proper* iff $\forall C \in S$: $\rho(C) \subseteq \{D \in S | C \succ D\}$
- The sets of *one-step refinements*, *n-step refinements* and *refinements* of a clause $C \in S$ are defined respectively as
  $\rho^1(C) = \rho(C)$
  $\rho^n(C) = \{D | \exists E \in \rho^{n-1}(C) \text{ and } D \in \rho(E)\}$
  $\rho^*(C) = \rho^1(C) \cup \rho^2(C) \cup \ldots \cup \rho^i(C) \cup \ldots$
- $\rho$ is called *complete* iff $\forall C, D \in S$ if $C \succ D$ then $\exists E \in \rho^*(C)$ and $E \sim D$[4].

## 3 Complete and Proper Refinement

Before we present our nonexistence results we motivate our interest in complete and proper refinement operators for an unrestricted search space. First of all, it should be clear that any refinement operator that is not locally finite is of no practical use. All refinement operators in this article will be locally finite.

Completeness of refinement operators is an important property since without it it is hard to make any statement concerning the performance of the systems in which they are used. Properness is a nice property for reasons of efficiency. If a clause is refuted because it is too general or too specific then we are not interested in clauses that are equivalent with this refuted clause. These clauses will also be too general or too specific. Still some refinement operators do return equivalent clauses. It will appear that improper refinement steps are sometimes necessary as a bridge to reach proper specializations or generalizations.

Since the clauses in the theory to learn are not known in advance, any restriction to the search space might exclude these clauses. Shapiro [15] solves this problem by incrementally expanding the search space. This, however, brings a lot of extra work. Furthermore, even if a clause and a proper specialization of it are both in a restricted search space, problems with finding a refinement chain between them can still occur if intermediate clauses are not in this search space.

[4] Our notion of completeness is stronger than Shapiro's [15] notion of completeness that is defined for downward refinement operators only by $\rho^*(\square) = S$, where $\square$ denotes the empty clause.

### 3.1 Nonexistence Conditions

*Example 2.* Consider the clauses

$D_2 = q(X_1) \leftarrow p(X_1, X_2), p(X_2, X_1)$
$D_n = q(X_1) \leftarrow p(X_1, X_2), p(X_2, X_1) \ldots, p(X_{n-1}, X_n), p(X_n, X_{n-1})$
$C = q(X_1) \leftarrow p(X_1, X_1)$

$D_n$ contains every literal $p(X_i, X_j)$, where $1 \leq i, j \leq n$ and $i \neq j$. In the following subsection we will show that if these clauses are ordered by $\theta$-subsumption they satisfy $D_2 \succ D_3 \succ \ldots \succ D_n \succ D_{n+1} \succ \ldots \succ C$, and no clause $E$ satisfies $D_n \succeq E \succ C$ for all $n$. Lemma 2 states for ordered search spaces that contain clauses like these, no locally finite, complete and proper upward refinement operator exists. The problems are illustrated in Figure 1b. The arrows illustrate proper generalization relations. Using a locally finite and proper upward refinement operator, all filled dots can be derived from $C$, whereas the open dots can not. Hence it can not be complete.

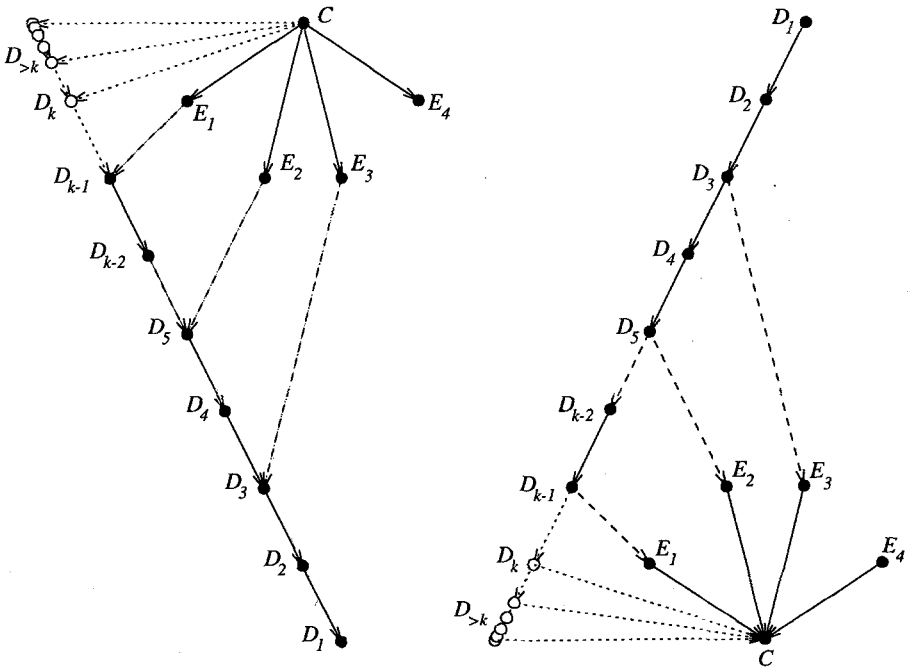


**Fig. 1.** Nonexistence conditions for locally finite, complete and proper refinement operators.

The following lemma states sufficient condition to conclude that locally finite, complete and proper downward refinement operator does not exist.

**Lemma 1.** *Let $S$ be ordered by $\succeq$. If $S$ contains clauses $C$ and $D_n$, $n \geq 1$ such that*

1. $C \succ \ldots \succ D_{n+1} \succ D_n \succ \ldots \succ D_2 \succ D_1$, *and*
2. $\nexists E$ *such that for all* $n \geq 1$: $C \succ E \succeq D_n$.

*Then a locally finite, complete and proper downward refinement operator for $S$ ordered by $\succeq$ does not exist.*

*Proof.* Assume that such a $\rho$ exists. Let $\rho(C) = \{E_1, \ldots, E_m\}$, then $C \succ E_i, 1 \leq i \leq m$. For every $E_i$, let $n_i = \min\{n | E_i \not\succeq D_n\}$. Because of condition 2 these $n_i$'s exist. Let $k = \max\{n_i\}$. Then $C \succ D_k$ and $E_i \not\succeq D_k$, $1 \leq i \leq m$. Thus, $D_k$ is not in $\rho(C)$ itself nor is $D_k$ derivable from any $E_i$. We conclude that $\rho$ is not complete. □

An analogous proof holds for the upward version of Lemma 1:

**Lemma 2.** *Let $S$ be ordered by $\succeq$. If $S$ contains clauses $D_n$, $n \geq 1$ and $C$ such that*

1. $D_1 \succ D_2 \succ \ldots \succ D_n \succ D_{n+1} \succ \ldots \succ C$, *and*
2. $\nexists E$ *such that for all* $n \geq 1$: $D_n \succeq E \succ C$.

*Then a locally finite, complete and proper upward refinement operator for $S$ ordered by $\succeq$ does not exist.*

We will apply these lemma's that are valid for arbitrary ordered search spaces to unrestricted search spaces ordered by $\theta$-subsumption. Example clauses that fit the lemma's can already be found in a logical language with one binary predicate $p$ and no function symbols. The nonexistence results of the succeeding subsections are valid for any logical language that contains infinitely many variables, one or more predicate symbols of arity> 1 and any number of function symbols.

For simplicity we use only positive literals in the construction of our example clauses. By changing these examples a little they can be transformed to program clauses. For example, if we use $\{p(X_1, X_2), p(X_2, X_1)\}$ then the same problems occur with the program clause $p(a, a) \leftarrow p(X_1, X_2), p(X_2, X_1)$.

### 3.2 Nonexistence for Upward Refinement

Throughout this subsection we use the clauses with the following underlying sets:

$\dot{K}_n = \bigcup\{p(X_i, X_j) | 1 \leq i, j \leq n, i \neq j\}$
$\dot{C} = \{p(X_1, X_1)\}$

$\dot{K}_n$ represents a structure that is known as a complete graph of size $n$, for example

$\dot{K}_3 = \{p(X_1, X_2), p(X_1, X_3), p(X_2, X_1), p(X_2, X_3), p(X_3, X_1), p(X_3, X_2)\}$.

Throughout this section $\dot{C}$ will be used whenever set properties are used, such as in $\dot{C} \subseteq \dot{D}$. Otherwise we write $C$, where $C$ is a clause that contains one occurrence of every literal in $\dot{C}$. Until Section 5 the difference between the two notations is not important and can be ignored.

We adopt from Plotkin's definition of reducedness [13]. A clause $C$ is called *reduced* iff $\dot{D} \subseteq \dot{C}$ and $D \sim C$ imply $\dot{C} = \dot{D}$. In words, $C$ is reduced iff it equivalent to no proper subset of itself.

**Lemma 3.** *For all $n \geq 2$, $K_n$ is reduced.*

*Proof.* Assume $K_n$ is not reduced for some $n$. Then for some substitution $\theta$, $\dot{K}_n\theta \subset \dot{K}_n$. 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 $\dot{K}_n\theta \subset \dot{K}_n$. □

**Lemma 4.** $K_2 \succ K_3 \succ \ldots \succ K_n \succ K_{n+1} \succ \ldots \succ C$.

*Proof.* For every $K_n$ we can define a $\theta$ that maps every $X_i$ in $K_n$ to $X_1$. This gives $\dot{K}_n\theta \subseteq \dot{C}$. Since $p(X_1, X_1)$ in $C$ cannot be mapped to any literal in any $K_n$ we get $K_n \succ C$.

Using the trivial substitution we can prove $K_n \succeq K_{n+1}$. Since $K_{n+1}$ is reduced (Lemma 3) and $\dot{K}_n \subset \dot{K}_{n+1}$, $K_{n+1}$ and $K_n$ cannot be equivalent, and $K_n \succ K_{n+1}$. □

**Lemma 5.** *Let $C$ and $K_n$ be defined as above. Then there is no $E$ such that for all $n \geq 2$, $K_n \succeq E \succ C$.*

*Proof.* Assume that $E$ satisfies $K_n \succeq E \succ C$ for all $n \geq 2$. Let $X_1, \ldots, X_m$ be all variables in $E$. By $\dot{E}\theta \subseteq \dot{C}$, $E$ can contain only literals $p(X_i, X_j)$. In these literals $X_i \neq X_j$ must hold, otherwise $E$ is equivalent with $C$. But then $\dot{E} \subseteq \dot{K}_m$ which implies $E \succeq K_m \succ K_{m+1}$. This contradicts $K_n \succeq E$ for $n = m + 1$. □

**Theorem 6.** *A locally finite, complete and proper upward refinement operator for unrestricted search spaces ordered by $\theta$-subsumption does not exist.*

*Proof.* Follows directly from Lemma 2, Lemma 4 and Lemma 5. □

### 3.3 Nonexistence for Downward Refinement

Throughout this subsection clauses with the following underlying sets are used:

$\dot{C} = \{p(X_1, X_2), p(X_2, X_1)\}$
$\dot{C}_n = \{p(Y_1, Y_2), p(Y_2, Y_3), \ldots, p(Y_{n-1}, Y_n), p(Y_n, Y_1)\}$
$\dot{D}_n = \dot{C} \cup \dot{C}_{3^n}$, $n \geq 1$

We state without proof that

**Lemma 7.** *For all $n \geq 1$, $D_n$ is reduced.*

**Lemma 8.** *Let $C$ and $D_n$ be defined as above. Then $C \succ \ldots \succ D_{n+1} \succ D_n \succ \ldots \succ D_2 \succ D_1$.*

*Proof.* $C \succ D_n$ follows directly from $\dot{C} \subset \dot{D}_n$ and the reducedness of $D_n$ (Lemma 7). Let $\theta$ be the substitution that maps every $Y_j$, $1 \leq j \leq 3^{n+1}$, in $D_{n+1}$ to $Y_k$ in $D_n$, where $k = 3^n$ iff $j \bmod 3^n = 0$ and $k = j \bmod 3^n$ otherwise. Then $\dot{D}_{n+1}\theta = \dot{D}_n$, and hence $D_{n+1} \succeq D_n$. Assume $D_n \succeq D_{n+1}$. Then for some $\sigma$, $\dot{D}_n\sigma \subseteq \dot{D}_{n+1}$ and since $|\dot{D}_n| < |\dot{D}_{n+1}|$, $\dot{D}_n\sigma \subset \dot{D}_{n+1}$. But then $\dot{D}_{n+1}\theta\sigma = \dot{D}_n\sigma \subset \dot{D}_{n+1}$, which contradicts that $D_{n+1}$ is reduced (Lemma 7). We conclude $D_n \not\succeq D_{n+1}$, and hence $D_{n+1} \succ D_n$. □

**Lemma 9.** *Let $C$ and $D_n$ be defined as above. Then there is no $E$ such that for all $n \geq 1$, $C \succ E \succeq D_n$.*

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

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

**Theorem 10.** *A locally finite, complete and proper downward refinement operator for unrestricted search spaces ordered by $\theta$-subsumption does not exist.*

*Proof.* Follows directly from Lemma 1, Lemma 8 and Lemma 9. □

## 4 Complete Downward Refinement

Laird has presented a generalized version of Shapiro's [15] refinement operator for reduced clauses in [7], where he referred to Shapiro's (incorrect) proof of (weak) completeness. We repeat the definition of Laird's downward refinement operator in our notation:

**Refinement operator $\rho_0$.** Let $C = L_1, \ldots, L_m \leftarrow M_1, \ldots, M_n$ be a clause. Then $D \in \rho_0(C)$ when exactly one of the following holds:

1. $D = C\theta$, where $\theta = \{X/Y\}$ and both variables $X$ and $Y$ occur in $C$.
2. $D = C\theta$, where $\theta = \{X/f(Y_1, \ldots, Y_n)\}$, $f$ is an $n$-ary function symbol, $X$ occurs in $C$ and $Y_1, \ldots, Y_n$ are distinct variables not occurring in $C$.
3. $D = L_1, \ldots, L_{m+1} \leftarrow M_1, \ldots, M_n$, where $L_{m+1}$ is a most general atom w.r.t. $C$.
4. $D = L_1, \ldots, L_m \leftarrow M_1, \ldots, M_{n+1}$, where $M_{n+1}$ is a most general atom w.r.t. $C$.

Where Shapiro needed to restrict the search space to a finite set, Laird's $\rho_0$ operates on unrestricted search spaces.

**Theorem 11.** *$\rho_0$ is a locally finite, complete but improper refinement operator for unrestricted search spaces ordered by $\theta$-subsumption.*

*Proof.* Local finiteness of $\rho_0$ follows directly from the finite number of variables in a clause and the definition of $\rho_0$. A proof of completeness can be found in [4]. Improperness is easy to verify. Consider for example the clauses $C = p(f(X)) \leftarrow p(X)$ and $D = p(f(X)) \leftarrow p(X), p(Y)$. They satisfy $C \sim D$, and $D \in \rho_0(C)$ by item 4. □

# 5 Complete Upward Refinement

Our intention is to define an upward refinement operator $\delta_0$ with the same properties as $\rho_0$. The different problems involved in the definition of such an upward refinement operator arise in different weaker orderings. We therefore consider three increasingly weak orderings: the $\theta$-subsumption, set, and substitution ordering, denoted by $\succeq, \succeq_1$, and $\succeq_2$ respectively. The corresponding complete upward refinement operators will be denoted by $\delta_0, \delta_1, \delta_2$. Since refinement operators for weaker and simpler orderings can be used to define refinement operators for stronger, more complex orderings, $\delta_0$, $\delta_1$ and $\delta_2$ will be defined in reverse order. The weak to strong approach was also used in [12], where we investigated proper refinement in restricted, finite search spaces. $\delta_1$ and $\delta_2$ that will be defined later on could already be found in that article.

## 5.1 The Substitution Ordering

In the *substitution ordering* $\succeq_2$, clauses are treated as sequences of literals, the number of the literals and their position in a clause are fixed. It is defined by $C \succeq_2 D$ iff $\exists\theta : C\theta = D$

*Example 3.* Consider

$C = even(X) \leftarrow odd(Y), plus(Y, Y, X)$ and
$D = even(Z) \leftarrow plus(3, 3, Z), odd(3)$.

$C$ and $D$ are incomparable in the substitution ordering because no substitution can map $odd(Y)$ to $plus(3, 3, Z)$ or the other way around. If the places of the body literals in either $C$ or $D$ were swapped then $C \succeq_2 D$ would hold by $\theta = \{X/Z, Y/3\}$ (in fact, $C \succ_2 D$).

In the substitution ordering, substitutions that are not renamings determine proper refinements [12]. In all clauses comparable with a clause $C$, predicate symbols appear in the same place as in $C$ and no literals can be removed or added. We can speak of clauses being treated as atoms, $L_1, \ldots, L_m \leftarrow M_1, \ldots, M_n$ can be viewed as $\vee(L_1, \ldots, L_m, \neg M_1, \ldots, \neg M_n)$ where the ordering of the arguments of $\vee$ is fixed. Reynolds [14] has described a (downward) cover relation for atoms which corresponds with items 1 and 2 in the definition of $\rho_0$. This relation can be used as a downward refinement operator for clauses w.r.t. the substitution ordering directly.

Our first upward refinement operator is obtained by inverting these substitutions. The dual of item 2 in $\rho_0$ has to be described seperately for constants and function symbols of arity $> 0$. Replacing some or all occurences of a constant $c$ by a new variable $X$ always inverts a $\rho_0$-substitution $\{X/c\}$. Replacing some but not all occurences $f(Y_1, \ldots, Y_n)$ by a variable $X$ does not invert a $\rho_0$-subsitition $\{X/f(Y_1, \ldots, Y_n)\}$ since it results in a clause in which $Y_i$ still occurs.

*Refinement operator* $\delta_2$. Let $C$ be a clause, then

$D \in \delta_2(C)$ iff one of the following holds:

1. $D$ is $C$ after some (not all) occurrences of a variable $Y$ in $C$ are replaced by a variable $X$ not in $C$.
2. $D$ is $C$ after all occurrences of a term $f(Y_1, \ldots, Y_n)$ are replaced by a variable $X$, where $f$ is a $n$-ary function symbol ($n > 0$), $X$ does not occur in $C$ and all $Y_i$'s are distinct variables not occurring elsewhere in $C$ besides in terms $f(Y_1, \ldots, Y_n)$.
3. $D$ is $C$ after some or all occurrences of a constant $c$ are replaced by a variable $X$, where $X$ does not occur in $C$.

**Lemma 12.** *$\delta_2$ is a locally finite refinement operator.*

*Proof.* Every clause contains a finite number of term occurrences. Therefore the number of possible inverse substitutions is finite and $\delta_2$ is locally finite. □

**Lemma 13.** *$\delta_2$ is a complete refinement operator for unrestricted search spaces w.r.t. the substitution ordering.*

*Proof.* It is proved by Reynolds [14, Theorem 4] that for every pair of atoms $A$ and $B$, if $A \succ_2 B$ then there is a finite chain $A = A_0, \ldots, A_n = B$ such that $A_i$ is a downward cover of $A_{i-1}$ and $A_i$ can be derived from $A_{i-1}$ through item 1 or 2 of $\rho_0$. Since $A$ is a downward cover of $B$ iff $B$ is an upward cover of $A$ we can use $\delta_2$ in the upward case. Since clauses are treated as atoms, this result can be generalized to unrestricted search spaces of clauses ordered by $\succeq_2$. □

### 5.2 The Set Ordering

In the *set ordering* $\succeq_1$, permutation of literals and addition or removal of duplicate literals in a clause no longer influence generality relations. The set ordering is strictly stronger than the substitution ordering [12]. It is defined by $C \succeq_1 D$ iff $\exists\theta : \dot{C}\theta = \dot{D}$.

*Set reduction* of a clause is the removal of all duplicate literals in it. A clause is *set reduced* iff it contains no duplicate literals. If set reduction of $C$ results in $C'$, then clearly $C \sim_1 C'$. We might therefore call $C'$ the set reduced equivalent of $C$.

In the set ordering, the necessity of adding literals in generalization steps arises:

*Example 4.* We repeat the clauses of Example 1:

$D = cycle(X) \leftarrow con(X, X)$
$D' = cycle(X) \leftarrow con(X, X), con(X, X), con(X, X)$
$E = cycle(X) \leftarrow con(X, X), con(X, Z), con(Z, X)$
$C = cycle(X) \leftarrow con(X, Y), con(Y, Z), con(Z, X)$

In the set ordering, $C \succ_1 E \succ_1 D' \sim_1 D$. These clauses illustrate a $\rho_0$-chain of downward refinement steps, $E \in \rho_0(C), D \in \rho_0(E)$. At every step two variables are unified and duplicate literals are removed. In the case of upward refinement, $C \in \delta_2(E)$ holds. However, $\delta_2$ can not be used to derive $E$ from $D$, since the number of literals must increase. If we duplicate $con(X, X)$ twice in $D$ before applying $\delta_2$, then we would obtain $D'$ and $E \in \delta_2(D')$. This motivates our definition of $\delta_1$ later on.

In the case of downward refinement operators, equal literals are of no use since they remain equal after substitution. Hence there is no need to duplicate literals at any time and clauses can be set reduced as soon as duplicate literals appear.

As the last example showed for the case of upward refinement, literals sometimes should be repeated before inverse substitutions are applied by $\delta_2$. We can easily define an operator that duplicates a literal:

Let $C = L_1, \ldots, L_m \leftarrow M_1, \ldots, M_n$ be a clause, then $D \in eq_1(C)$ iff
$D = L_1, \ldots, L_m \leftarrow M_1, \ldots, M_i, M_i, \ldots, M_n$ or
$D = L_1, \ldots, L_i, L_i, \ldots, L_m \leftarrow M_1, \ldots, M_n$.
By applying $eq_1$ zero or more times, $eq_1^*(C)$ contains infinitely many clauses of the form $L_1, \ldots, L_1, L_2, \ldots, L_2, \ldots, L_m, \ldots, L_m \leftarrow$
$M_1, \ldots, M_1, M_2, \ldots, M_2, \ldots, M_n, \ldots, M_n$.

In his description of the inversion of $\theta$-subsumption Jung [3] also incorporated the addition of arbitrary many copies of body literals to a clause. Later on in this section we will show that only a finite part of $eq_1^*$ is needed for computing one-step upward refinements.

*Refinement operator* $\delta_1$. Let $C$ be a set reduced clause, then

$D \in \delta_1(C)$ iff there are $C' \in eq_1^*(C)$, $D \in \delta_2(C')$ and $D$ is set reduced.

Note that clauses with duplicate literals are hard to describe when clauses are represented as sets of literals. We therefore need the sequence of literals representation of clauses for clauses that are obtained by $eq_1^*$ and submitted to $\delta_2$.

The proof of the following lemma contains the solution for the main problem of defining a locally finite complete upward refinement operator for unrestricted search spaces ordered by $\succeq_1$ or $\succeq$. It shows that the number of necessary literal repetitions is finite and computable.

**Lemma 14.** *$\delta_1$ is a locally finite refinement operator.*

*Proof.* We show that, no matter how many times every single literal of $C$ is repeated before $\delta_2$ is applied, $\delta_1(C)$ contains finitely many nonequivalent clauses.

Given a clause $C$, if $C' \in eq_1^*(C)$ and $D \in \delta_2(C')$, then $D$ contains exactly one variable, say $X$, that is not in $C'$ and $C$. Suppose that $\delta_2$ replaces some or all of the occurrences of a constant $c$ in $C'$ by $X$ (the cases of anti-unification and functional terms can be proved similarly). Let $c$ occur in the literals $L_1, \ldots, L_m$ of $C$. In every such literal $L_i$, $c$ occurs finitely many times, say $n_i$ times. When $\delta_2$ is applied to $C'$ every single occurrence of $c$ is either replaced by $X$ or not. For a single literal $L_i$ this results in $2^{n_i}$ possible different literals. Hence there is no need to repeat $L_i$ more than $2^{n_i}$ times.

For each literal $L$ in $C$ we can thus compute an upper bound of the sufficient number of repetitions of $L$ in $C'$. Hence we only have to consider a finite and computable part of $eq_1^*(C)$ in the definition of $\delta_1$. □

The following example illustrates the local finiteness of $\delta_1$.

*Example 5.* Consider the clause

$$C = q(b) \leftarrow p(a,a)$$

We describe the case in which $\delta_2$ replaces some or all occurrences of $a$ by a new variable $X$. Then $L = p(a,a)$ can either become $p(X,X)$, $p(X,a)$, or $p(a,X)$, or remain unchanged. We claim that no more than $2^2 = 4$ occurrences of $L$ in $C'$ are useful. Consider

$$C' = q(b) \leftarrow p(a,a), p(a,a)$$
$$C'' = q(b) \leftarrow p(a,a), p(a,a), p(a,a), p(a,a)$$
$$D_1 = q(b) \leftarrow p(a,X), p(X,a)$$
$$D_2 = q(b) \leftarrow p(a,X), p(X,a), p(a,X), p(X,a)$$
$$D_3 = q(b) \leftarrow p(a,X), p(X,a), p(X,X), p(a,a)$$

Then $D_1$ is one of the clauses in $\delta_2(C')$ and $D_2$ and $D_3$ are two of the many possible clauses in $\delta_2(C'')$. $D_2$ contains the literals $p(a,X)$ and $p(X,a)$ twice and is not set reduced. $D_2$ is therefore not a member of $\delta_1(C)$.

Proper generalizations of $C$ with more than five literals do exist. But they all have at least two variables that are not in $C$. Consider for example the clause

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

then $E$ is not derivable from $C$ in one step. It is however derivable in two steps.

**Lemma 15.** *$\delta_1$ is a complete upward refinement operator for unrestricted search spaces w.r.t. the set ordering.*

*Proof.* Let $C$ and $D$ be set reduced clauses such that $D \succ_1 C$. Then there exists a substitution $\theta$ such that $\dot{D}\theta = \dot{C}$.

$D \succeq_2 D\theta$, so by the completeness of $\delta_2$ there exists a finite $\delta_2$-chain from $D\theta$ to a clause $D' \sim_2 D$.

$D\theta$ possibly differs from $C$ in the repetition of literals and a permutation of literals. Since in the definition of $\delta_1$ all necessary literals are repeated before $\delta_2$ is applied and the ordering of literals does not influence the applicability of $\delta_2$,

there exists a $\delta_1$-chain from $C$ to $D'' \sim_1 D$ containing all $\delta_2$ steps of the $\delta_2$-chain from $D\theta$ to $D'$, possibly preceded by literal duplications. □

*Example 6.* We will illustrate the proof of Lemma 5.7 using the following clauses:

$C = even(X) \leftarrow odd(3), plus(3,3,X)$
$D = even(U) \leftarrow plus(V,W,U), odd(V), odd(W)$

$\dot{D}\theta = \dot{C}$ by $\theta = \{U/X, V/3, W/3\}$. Hence there exists a $\delta_2$-chain from $D\theta$ to a clause $D' \sim_1 D$:

$D\theta = even(X) \leftarrow plus(3,3,X), odd(3), odd(3)$
$E = even(X) \leftarrow plus(Z,Z,X), odd(Z), odd(Z) \in \delta_2(D'\theta)$
$D' = even(X) \leftarrow plus(Z,W,X), odd(Z), odd(W) \in \delta_2(E)$

This $\delta_2$-chain from $D\theta$ to $D'$ can indeed be transformed to a $\delta_1$-chain from $C$ to $D''$ by adding the necessary literal duplications:

$C = even(X) \leftarrow odd(3), plus(3,3,X)$
$E' = even(X) \leftarrow odd(Z), plus(Z,Z,X) \in \delta_1(C)$
$E'' = even(X) \leftarrow odd(Z), odd(Z), plus(Z,Z,X) \in eq_1^*(E')$
$D'' = even(X) \leftarrow odd(Z), odd(W), plus(Z,W,X) \in \delta_2(E'')$ $(D'' \in \delta_1(E'))$

Note that every literal modification in the $\delta_2$-chain $D\theta, E, D'$ returns in the $\delta_1$-chain $C, E', D''$. The literal $odd(Z)$ is duplicated once in $E'$ in order to derive $D''$.

### 5.3 The $\theta$-Subsumption Ordering

The $\theta$-subsumption ordering is strictly stronger than the set ordering [12]. To transform our upward refinement operator for the set ordering ($\dot{C}\theta = \dot{D}$) to one for the $\theta$-subsumption ordering ($\dot{C}\theta \subseteq \dot{D}$) is relatively easy. We only have to incorporate an operation for removing literals. We therefore invert the operation of adding a most general literal:

**Refinement operator $\delta_0$.** Let $C$ be a set reduced clause, then

$D \in \delta_0(C)$ iff $D \in \delta_1(C)$ or
$D$ is $C$ after removing a literal that is most general w.r.t. $D$.

**Theorem 16.** *$\delta_0$ is a locally finite, complete but improper upward refinement operator for unrestricted search spaces ordered by $\theta$-subsumption.*

*Proof. Local finiteness.* Every clause $C$ contains finitely many literals, hence finitely many one-step refinements can be added by the new operation of removing a literal. Local finiteness of $\delta_0$ then follows from the local finiteness of $\delta_1$.
*Completeness (outline).* Let $D \succeq C$, then for some $\theta$, $\dot{D}\theta \subseteq \dot{C}$. First we prove that there exists a finite $\delta_0$-chain from $C$ to $D\theta$. Let $\{M_1, \ldots, M_n\} = \dot{C} - \dot{D}\theta$.

Let $\dot{C}_i = \dot{D}\theta \cup \{M_1, \ldots, M_i\}$, then $\dot{C}_n = \dot{C}$. We will show that we can successively derive $\dot{C}_{n-1}, \ldots, \dot{C}_0 = \dot{D}\theta$.

For $i = n - 1$ downto 1:
We start with $\dot{C}_{i+1} = \dot{C}_i \cup \{M_{i+1}\}$.
Let $L_{i+1}$ denote a literal that is compatible with $M_{i+1}$ and most general w.r.t. $\dot{C}_i$. Then $\dot{C}_{i+1} = (\dot{C}_i \cup \{L_{i+1}\})\theta$ for some $\theta$.
-By the completeness of $\delta_1$, we can derive $\dot{C}_i \cup \{L_{i+1}\}$ from $\dot{C}_{i+1}$.
-Since $L_{i+1}$ is most general w.r.t. $\dot{C}_i$, we can remove $L_{i+1}$ from $\dot{C}_i \cup \{L_{i+1}\}$, which results in $\dot{C}_i$.

Since there are finitely many literals $M_i$, all these operations can be performed in a finite number of refinement steps. We now have a finite $\delta_0$-chain from $C$ to $D\theta$. By the completeness of $\delta_1$, there exists a finite $\delta_1$-chain from $D\theta$ to $D' \sim D$. Since all operations of $\delta_1$ are operations of $\delta_0$ too, this $\delta_1$-chain can be used to complete the $\delta_0$-chain from $C$ to $D' \sim D$.
*Improperness.* Consider the clauses

$C = p(f(X)) \leftarrow p(X), p(Y)$
$D = p(f(X)) \leftarrow p(X)$

As can be verified, $C \sim D$, but $D \in \delta_0(C)$. □

## 6 Conclusions and Future Research

In this article we have presented some new results regarding refinement in unrestricted search spaces ordered by $\theta$-subsumption. A localy finite, complete but improper upward refinement operator has been defined, a downward refinement operator with these properties already existed. We have proven that locally finite downward and upward refinement operators that are both complete and proper for these search spaces do not exist.

Our current and future research involves a logical framework in which all refinement operators that we know of find their place. Operators are classified according to being upward or downward and properties such as local finiteness, properness, weak and strong completeness w.r.t. an (un)restricted search space and its generality ordering. Using this framework we try to fill the categories in which no refinement operators are known.

**Acknowledgment.** We thank Leon van der Torre for reading and commenting the article as well as for his moral support.

## References

1. P. Idestam-Almquist. Generalization under Implication by Using Or-Introduction. In P.B. Brazdil, editor, *ECML-93*, pages 56–64, Vienna, Austria, April 1993. LNAI-667, Springer-Verlag.
2. P. Idestam-Almquist. Recursive Anti-unification. In S. Muggleton, editor, *ILP'93*, pages 241–253, Bled, Slovenia, March 1993. Technical Report IJS-DP-6707, J.Stefan Institute.

3. B. Jung. On Inverting Generality Relations. In S. Muggleton, editor, *ILP'93*, pages 87–101, Bled, Slovenia, March 1993. Technical Report IJS-DP-6707, J.Stefan Institute.
4. P.R.J. van der Laag. Een Meest Algemene Verfijningsoperator voor Gereduceerde Zinnen. In *NAIC-92*, pages 29–39. Delftse Universitaire Pers, 1992. In Dutch, English version has appeared as Technical Report EUR-CS-92-03, Erasmus Univerity of Rotterdam, Dept. of Computer Science.
5. P.R.J. van der Laag and S.H. Nienhuys-Cheng. A Locally Finite and Complete Upward Refinement Operator for $\theta$-Subsumption. In *Benelearn-93*. Artificial Intelligence Laboratory, Vrije Universiteit Brussel, Brussels, 1993.
6. P.R.J. van der Laag and S.H. Nienhuys-Cheng. Subsumption and Refinement in Model Inference. In *ECML-93*, pages 95–114, Vienna, Austria, April 1993. LNAI-667, Springer Verlag.
7. P.D. Laird. *Learning from Good and Bad Data.* Kluwer Academic Publishers, 1988.
8. S. Lapointe and S. Matwin. Subunification: A Tool for Efficient Induction of Recursive Programs. In *ML-92*, pages 273–280, Aberdeen, 1992. Morgan Kaufmann.
9. C. Ling and M. Dawes. SIM the Inverse of Shapiro's MIS. Technical report, Department of Computer Science, University of Western Ontario, London, Ontario, Canada., 1990.
10. S.H. Muggleton. Inverting Implication. In Muggleton, S.H., editor, *Proceedings of the International Workshop on Inductive Logic Programming*, 1992.
11. T. Niblett. A Note on Refinement Operators. In *ECML-93*, pages 329–335. LNAI-667, Springer Verlag, 1993.
12. S.H. Nienhuys-Cheng, P.R.J. van der Laag, and L.W.N. van der Torre. Constructing Refinement Operators by Decomposing Logical Implication. In P. Torasso, editor, *AI*IA '93*, pages 178–189, Torino, Italy, October 1993. LNAI-728, Springer-Verlag.
13. G.D. Plotkin. A Note on Inductive Generalization. *Machine Intelligence*, 5:153–163, 1970.
14. J.C. Reynolds. Transformational Systems and the Algebraic Structure of Atomic Formulas. *Machine Intelligence*, 5:135–153, 1970.
15. E.Y. Shapiro. Inductive Inference of Theories from Facts. Technical Report 192, Department of Computer Science, Yale University, New Haven. CT., 1981.