A Note on Refinement Operators for IE-Based ILP Systems

Alireza Tamaddoni-Nezhad and Stephen Muggleton

Department of Computing, Imperial College London
{atn,shm}@doc.ic.ac.uk

Abstract. ILP systems which use some form of Inverse Entailment (IE) are based on clause refinement through a hypotheses space bounded by a most specific clause. In this paper we give a new analysis of refinement operators in this setting. In particular, Progol's refinement operator is revisited and discussed. It is known that Progol's refinement operator is incomplete with respect to the general subsumption order. We introduce a subsumption order relative to a most specific (bottom) clause. This subsumption order, unlike previously suggested orders, characterises Progol's refinement space. We study the properties of this subsumption order and show that ideal refinement operators exist for this order. It is shown that efficient operators can be implemented for least generalisation and greatest specialisation in the subsumption order relative to a bottom clause. We also study less restricted subsumption orders relative to a bottom clause and show how Progol's incompleteness can be addressed.

1 Introduction

Searching a refinement lattice bounded below by a bottom clause is the basis of several state-of-the-art ILP systems (e.g. Progol [7], Aleph [11]). These systems use refinement operators together with a search method to explore a bounded hypotheses space. For example, in the default setting, Progol uses a A*-like search together with a top-down refinement operator. Progol's refinement operator is designed to avoid redundancy in a A*-like search. This, however, leads to incompleteness of Progol's refinement operator with respect to the general subsumption order. There have been previous attempts to characterise Progol's refinement space and also to address the Progol's incompleteness. In particular, special cases of subsumption have been suggested to characterise Progol's refinement space [1]. In this paper we give a new analysis of refinement operators in a Progol-like ILP system. We introduce a subsumption order relative to a most specific (bottom) clause. This subsumption order, unlike previously suggested orders, characterises Progol's refinement space. We show the existence of ideal refinement operators for this subsumption order. It is also shown that efficient operators can be designed for least generalisation and greatest specialisation in the subsumption order relative to a bottom clause. We also study less restricted subsumption orders relative to a bottom clause and show how Progol's incompleteness (due to the choice of ordering in the bottom clause) can be addressed.

The theoretical results presented in this paper can be applied to ILP systems which use Inverse Entailment (IE) as well as any other ILP system which uses a most specific clause to restrict the search space. In this paper we first review clause refinement in Progol as an example of an IE-based ILP system. Section 3, gives examples of Progol's incompleteness with respect to the general subsumption order. In order to characterise Progol's refinement, Sections 4 defines sequential subsumption order relative to a bottom clause and describes the properties of this subsumption order. In section 5, refinement operators are defined for subsumption order relative to a bottom clause and the properties of these operators are discussed. Section 6 describes less restricted subsumption orders relative to a bottom clause and shows how Progol's incompleteness can be addressed. Related work is discussed in Section 7. Section 8 concludes the paper.

2 Clause Refinement in Progol

We assume the reader to be familiar with the basic concepts from logic programming and inductive logic programming [8] and also the basic concepts from ordered sets and lattices [3]. The following definition is a reminder of the concept of refinement operators and several properties of these operators.

Definition 1 ([8,1]). *Let $\langle G, \succeq \rangle$ be a quasi-ordered set. A (downward) refinement operator for $\langle G, \succeq \rangle$ is a function ρ, such that $\rho(C) \subseteq \{D|C \succeq D\}$, for every $C \in G$.*

- *The sets of one-step refinements, n-step refinements and refinements of some $C \in G$ are respectively: $\rho^1(C) = \rho(C)$, $\rho^n(C) = \{D|$ there is an $E \in \rho^{n-1}(C)$ such that $D \in \rho(E)\}, n \geq 2$ and $\rho^*(C) = \rho^0(C) \cup \rho^1(C) \cup ..$*
- *A ρ-chain from C to D is a sequence $C = C_0, C_1, \ldots, C_n = D$, such that $C_i \in \rho(C_i - 1)$ for every $1 \leq i \leq n$.*
- *ρ is locally finite if for every $C \in G$, $\rho(C)$ is finite and computable.*
- *ρ is proper if for every $C \in G$, $\rho(C) \subseteq \{D|C \succ D\}$.*
- *ρ is is complete if for every $C, D \in G$ such that $C \succ D$, there is an $E \in \rho^*(C)$ such that $D \sim E$ (i.e. D and E are equivalent in the $\succeq$-order).*
- *ρ is weakly complete if $\rho^*(\square) = G$, where $\square$ is the top element of G.*
- *ρ is non-redundant if for every $C, D, E \in G$, $E \in \rho^*(C)$ and $E \in \rho^*(D)$ implies $C \in \rho^*(D)$ or $D \in \rho^*(C)$.*
- *ρ is ideal if it is locally finite, proper and complete.*
- *ρ is optimal if it is locally finite, non-redundant and weakly complete.*

We can define analogous concepts for the dual case of an upward refinement operator.

The Progol algorithm [7] is based on successive construction of definite clause hypotheses H from a language $\mathcal{L}$. H must explain the examples E in terms of background knowledge B. Each clause in H is found by choosing an uncovered positive example e and searching through the graph defined by the refinement ordering $\succeq$ bounded below by the bottom clause associated with e. We define this setting more formally as follows.

Definition 2 (Progol refinement setting). *Let $\mathcal{S} = \langle B, E, \mathcal{L}, \succeq \rangle$ be Progol's ILP setting as defined in [7]. Let E consist of a set of positive and negative examples (ground unit clauses). The empty clause, denoted by $\square$, is the maximal$_{\succeq,\mathcal{L}}$ element in $\mathcal{L}$. The "bottom" clause, denoted by $\bot$, is the least$_{\succeq,\mathcal{L}}$ element such that $B, \bot \models e$. Refinement of clause C, denoted by $\rho(C)$, is the set of maximal$_{\succeq,\mathcal{L}}$ clauses D such that $C \succ D \succeq \bot$.*

The refinement operator in Progol is designed to avoid redundancy and to maintain the relationship $\square \succeq H \succeq \bot$ for each clause H. Since $H \succeq \bot$, it is the case that there exists a substitution θ such that $H\theta \subseteq \bot$. Thus for each literal l in H there exists a literal l' in $\bot$ such that $l\theta = l'$. Clearly there is a uniquely defined subset $\bot(H)$ consisting of all l' in $\bot$ for which there exists l in H and $l\theta = l'$. A non-deterministic approach to choosing an arbitrary subset S' of a set S involves maintaining an index k. For each value of k between 1 and n, the cardinality of S, we decide whether to include the kth element of S in S'. Clearly, the set of all series of n choices corresponds to the set of all subsets of S. Also for each subset of S there is exactly one series of n choices. To avoid redundancy and maintain θ-subsumption of $\bot$ Progol's refinement operator maintains both k and θ. The refinement operator ρ defined in [7] allows more than one literal in H to be mapped to the same literal l' in $\bot$. However, in Progol's implementation of the refinement operator, index k is incremented after each step for the sake of efficiency. This means each literal of $\bot$ can be considered only once. In Appendix A, we give a revised definition (ρ_0) which describes the refinement operator as implemented in Progol.

3 Characterising Progol's Refinement

In this section we show that Progol's refinement cannot be described by the general subsumption order and that we need the notion of "sequential subsumption" in order to capture Progol's refinement. It can be shown that a refinement operator cannot be both complete and non-redundant [1]. However, a refinement operator can be weakly complete and non-redundant (optimal). As mentioned in the previous section, Progol's ρ is designed to be non-redundant and therefore it cannot be complete. However, it is known that Progol's refinement operator is also not weakly complete with respect to the general subsumption order [7]. This is demonstrated in the following example[1].

Example 1. Let B contain definitions for decrementation (dec), addition (plus) and the clause $mult(0, X, 0) \leftarrow$ with appropriate mode declarations M and let the example e be the clause $mult(1, 1, 1) \leftarrow$. Then $\bot$ is the clause

$$\begin{aligned} \text{mult}(A, A, A) \leftarrow\ & \text{dec}(A, B), \text{plus}(B, A, A), \text{plus}(B, B, B), \\ & \text{mult}(B, A, B), \text{mult}(B, B, B). \end{aligned}$$

[1] This example is a corrected version of Example 30 in [7].

Now consider clause C:

$$C = \mathrm{mult}(U, V, W) \leftarrow \mathrm{dec}(U, X), \mathrm{mult}(X, V, Y), \mathrm{plus}(Y, V, W).$$

Clause C is in $\mathcal{L}$, but given the ordering over $\perp$ there will be no element of Progol's $\rho^*(\Box)$ containing this clause or a subsume-equivalent of this clause. $\diamond$

This first type of incompleteness is due to the choice of ordering in the bottom clause and the variable dependencies in the literals. As mentioned in the previous section, Progol's refinement uses an indexing over the literals and the literals in $\perp$ can only be considered from left to right.

As mentioned in the previous section, each literal from $\perp$ can be selected only once. This leads to the second type of incompleteness. The example below shows that Progol's refinement space is not a lattice with respect to the general subsumption, as the least general generalisation of clauses is not always in the refinement space.

Example 2. Let C, D and $\perp$ be clauses as defined below

$$\begin{aligned} C &= \mathrm{p}(X, Y) \leftarrow \mathrm{q}(X, X), \mathrm{q}(Y, W). \\ D &= \mathrm{p}(X, Y) \leftarrow \mathrm{q}(Z, X), \mathrm{q}(Y, Y). \\ \perp &= \mathrm{p}(X, Y) \leftarrow \mathrm{q}(X, X), \mathrm{q}(Y, Y). \end{aligned}$$

C and D can be generated by Progol's refinement given $\perp$, however, clause E below which is the least general generalisation (*lgg*) of C and D cannot be generated.

$$E = \mathrm{p}(X, Y) \leftarrow \mathrm{q}(Z, X), \mathrm{q}(U, U), \mathrm{q}(Y, W). \qquad \diamond$$

As another example of the second type of incompleteness, consider the following example adopted from [1].

Example 3. Let $\perp = \mathrm{p}(X) \leftarrow \mathrm{q}(X, X)$, then Progol's refinement only considers the following hypotheses.

$$\begin{aligned} C_1 &= \mathrm{p}(X) \\ C_2 &= \mathrm{p}(X) \leftarrow \mathrm{q}(X, X) \\ C_3 &= \mathrm{p}(X) \leftarrow \mathrm{q}(X, Y) \end{aligned}$$

However, the following clauses which subsume $\perp$ are not considered by Progol's refinement.

$$\begin{aligned} C'_1 &= \mathrm{p}(X) \leftarrow \mathrm{q}(X, Y), \mathrm{q}(Y, X) \\ C'_2 &= \mathrm{p}(X) \leftarrow \mathrm{q}(X, Y), \mathrm{q}(Y, Z), \mathrm{q}(Z, X) \\ &\ldots \end{aligned} \qquad \diamond$$

It has been suggested [1] that the second type of incompleteness is not a drawback as it can be justified by the examples and the MDL heuristic. In order to characterise Progol's refinement, the authors of [1] suggested a special case of subsumption, called weak subsumption, which does not allow substitutions that identify literals (i.e. for $C\theta$ there are no literals L_1 and L_2 in C such that $L_1\theta = L_2\theta$). For example, the clause $\mathrm{p}(X') \leftarrow \mathrm{q}(X', Y'), \mathrm{q}(Y', X')$ subsumes $\bot = \mathrm{p}(X) \leftarrow \mathrm{q}(X, X)$ with respect to the general subsumption, but it does not weakly subsume it. This is because substitution $\{X'/X, Y'/X\}$ identifies literals $\mathrm{q}(X', Y')$ and $\mathrm{q}(Y', X')$. The weak subsumption ordering, therefore, characterises the second type of incompleteness. However, it does not capture the incompleteness due to the ordering of the literals. For example, consider clauses C and $\bot$ in Example 1. C weakly subsumes $\bot$ but clause C is not considered by Progol's refinement.

As mentioned in the previous section, Progol's refinement operator scans $\bot$ from left to right and for each literal l' of $\bot$ decides whether to include a generalisation of it (i.e. l, where $l\theta = l'$) in H or not. $H\theta$ can be, therefore, characterised as a "subsequence" of $\bot$ rather than a "subset" of $\bot$. In the following sections we first define a special case of subsumption based on the idea of subsequences, and then we study the properties of this subsumption order. We show that Progol's refinement can be characterised by sequential subsumption relative to $\bot$. We also show that ideal refinement operators exist for this special case of subsumption.

4 Sequential Subsumption

Even though Progol does not use an explicit representation for ordered clauses, clauses in $\mathcal{L}$ are defined with a total ordering over the literals (Definition 21 in [7]). In order to characterise Progol's refinement we adopt an explicit representation for ordered clauses. The idea of ordered (or sequential) clauses is to consider a clause as a sequence of literals rather than a set of literals. This idea has been used in logic programming, in particular in the context of resolution (e.g. [2]). The concept of ordered clauses has been also used in ILP. For example, when defining upward refinement operators it is sometime necessary to duplicate literals in order to correctly invert an elementary substitution. Duplication of literals is not allowed for conventional clauses (which use a set notation) and therefore ordered clauses are used instead [8]. A subsumption relation for ordered clauses is studied in [4]. The difference between this subsumption order and the subsumption order considered in this paper is discussed in Section 7. There are also other applications of ordered clauses, for example in the context of data mining from sequential data(e.g. [6]). In this paper we use the same notion used in [8] and an ordered clause is represented as a disjunction of literals (i.e. $L_1 \vee L_2 \vee \cdots \vee L_n$). The set notation (i.e. $\{L_1, L_2, \ldots, L_n\}$) is used to represent conventional clauses.

Definition 3 (Ordered clause). *An ordered clause $\overrightarrow{C}$ is a sequence of literals $L_1, L_2, \ldots, L_n$ and denoted by $\overrightarrow{C} = L_1 \vee L_2 \vee \cdots \vee L_n$. The set of literals in $\overrightarrow{C}$ is denoted by C.*

Unlike conventional clauses, the order and duplication of literals matter for ordered clauses. For example, $\overrightarrow{C} = p(X) \vee \neg q(X)$, $\overrightarrow{D} = \neg q(X) \vee p(X)$ and $\overrightarrow{E} = p(X) \vee \neg q(X) \vee p(X)$ are different ordered clauses while they all correspond to the same conventional clause, i.e. $C = D = E = \{p(X), \neg q(X)\}$.

Selection of two clauses is defined as a pair of compatible literals and this concept was used by Plotkin to define least generalisation for clauses [9]. However, in this paper we use selections to define mappings of literals between two ordered clauses.

Definition 4 (Compatible literals). *Literals L and M are compatible if they have the same sign and predicate symbol.*

Definition 5 (Selection of clauses). *Let $\overrightarrow{C} = L_1 \vee L_2 \vee \cdots \vee L_n$ and $\overrightarrow{D} = M_1 \vee M_2 \vee \cdots \vee M_m$ be ordered clauses. A selection of $\overrightarrow{C}$ and $\overrightarrow{D}$ is a pair (i, j) where L_i and M_j are compatible literals.*

Definition 6 (Selection function). *Let $\overrightarrow{C} = L_1 \vee L_2 \vee \cdots \vee L_n$ and $\overrightarrow{D} = M_1 \vee M_2 \vee \cdots \vee M_m$ be ordered clauses. A set s of selections of $\overrightarrow{C}$ and $\overrightarrow{D}$ is called a selection function if it is a total function of $\{1, 2, \ldots, n\}$ into $\{1, 2, \ldots, m\}$.*

Example 4. Let $\overrightarrow{C} = L_1 \vee L_2 \vee L_3$ and $\overrightarrow{D} = M_1 \vee M_2 \vee M_3 \vee M_4$ be two ordered clauses and the set of all selections of $\overrightarrow{C}$ and $\overrightarrow{D}$ be $S = \{(1,1), (1,2), (2,1), (2,2), (3,4)\}$. Then, $s_1 = \{(1,1), (2,2), (3,4)\}$, $s_2 = \{(1,1), (2,1), (3,4)\}$ and $s_3 = \{(1,2), (2,1), (3,4)\}$ are examples of selection functions of $\overrightarrow{C}$ and $\overrightarrow{D}$. ◇

Definition 7 (Subsequence). *Let $\overrightarrow{C} = L_1 \vee L_2 \vee \cdots \vee L_l$ and $\overrightarrow{D} = M_1 \vee M_2 \vee \cdots \vee M_m$ be ordered clauses. $\overrightarrow{C}$ is a subsequence of $\overrightarrow{D}$, denoted by $\overrightarrow{C} \sqsubseteq^s \overrightarrow{D}$, if there exists a strictly increasing selection function s such that for each $(i, j) \in s$, $L_i = M_j$.*

Example 5. Suppose that in Example 4, we have $L_1 = L_2 = M_1 = M_2$ and $L_3 = M_4$. $\overrightarrow{C}$ is then a subsequence of $\overrightarrow{D}$ because there exists strictly increasing selection function s_1 which maps each literal L_i from $\overrightarrow{C}$ to an equivalent literal M_j from $\overrightarrow{D}$. ◇

Definition 8 (Ordered substitution). *Let $\overrightarrow{C} = L_1 \vee L_2 \vee \cdots \vee L_l$ be an ordered clause and θ be a substitution. $\overrightarrow{C}\theta$ is defined as follows, $\overrightarrow{C}\theta = L_1\theta \vee L_2\theta \vee \cdots \vee L_l\theta$.*

Definition 9 (Sequential subsumption). *Let $\overrightarrow{C}$ and $\overrightarrow{D}$ be ordered clauses. We say $\overrightarrow{C}$ is a sequential generalisation of $\overrightarrow{D}$, denoted by $\overrightarrow{C} \succeq_s \overrightarrow{D}$, if there exists a substitution θ such that $\overrightarrow{C}\theta$ is a subsequence of $\overrightarrow{D}$. $\overrightarrow{C}$ is a proper sequential generalisation of $\overrightarrow{D}$, denoted by $\overrightarrow{C} \succ_s \overrightarrow{D}$, if $\overrightarrow{C} \succeq_s \overrightarrow{D}$ and $\overrightarrow{D} \not\succeq_s \overrightarrow{C}$. $\overrightarrow{C}$ and $\overrightarrow{D}$ are equivalent with respect to sequential subsumption, denoted by $\overrightarrow{C} \sim_s \overrightarrow{D}$, if $\overrightarrow{C} \succeq_s \overrightarrow{D}$ and $\overrightarrow{D} \succeq_s \overrightarrow{C}$.*

Example 6. Let $\overrightarrow{B} = p(X_1, Y_1) \vee q(X_1, Y_1) \vee r(X_1, Y_1) \vee r(Y_1, X_1)$, $\overrightarrow{C} = p(X_2, Y_2) \vee r(U_2, Y_2) \vee r(Y_2, V_2)$ and $\overrightarrow{D} = p(X_3, Y_3) \vee r(Y_3, V_3) \vee r(U_3, Y_3)$ be ordered clauses. Let $\theta_1 = \{X_2/X_1, Y_2/Y_1, U_2/X_1, V_2/X_1\}$, then $\overrightarrow{C}\theta_1$ is a subsequence of $\overrightarrow{B}$ and therefore $\overrightarrow{C} \succeq_s \overrightarrow{B}$. However, there is no substitution θ_2 such that $\overrightarrow{D}\theta_2$ is a subsequence of $\overrightarrow{B}$ and therefore $\overrightarrow{D} \not\succeq_s \overrightarrow{B}$. Note that for conventional clauses B, C and D we have $C\theta_1 \subseteq B$ and similarly for $\theta_2 = \{X_3/X_1, Y_3/Y_1, V_3/X_1, U_3/X_1\}$ we have $D\theta_2 \subseteq B$ and therefore $C \succeq B$ and $D \succeq B$. ◇

The following theorem shows the relationship between sequential subsumption and the general subsumption order.

Theorem 1. *Let $\overrightarrow{C}$ and $\overrightarrow{D}$ be ordered clauses. If $\overrightarrow{C} \succeq_s \overrightarrow{D}$, then $C \succeq D$.*

Proof. Suppose $\overrightarrow{C} \succeq_s \overrightarrow{D}$, then according to Definition 9 there exists a substitution θ such that $\overrightarrow{C}\theta$ is a subsequence of $\overrightarrow{D}$. Let $\overrightarrow{C}\theta = L_1\theta \vee L_2\theta \vee \cdots \vee L_l\theta$ and $\overrightarrow{D} = M_1 \vee M_2 \vee \cdots \vee M_m$. Then for every literal $L_i\theta$ in $\overrightarrow{C}\theta$ there exists a literal M_j in $\overrightarrow{D}$ such that $L_i\theta = M_j$, and therefore $C\theta \subseteq D$. Hence, $C \succeq D$. □

Note that as shown in Example 6, the converse of Theorem 1 does not hold in general.

The languages which we consider in this paper (e.g. $\mathcal{L}$, $\overrightarrow{\mathcal{L}}_\perp$, etc.) correspond to a set of clauses which are generalisations of a flattened bottom clause [7]. Therefore, all clauses in $\mathcal{L}$, $\overrightarrow{\mathcal{L}}_\perp$, etc. are function-free and all substitutions we consider are variable substitutions[2]. Progol's refinement considers a subset of clauses in $\mathcal{L}$ which are sequential generalisation of the bottom clause. This subset of ordered clauses are defined as follows.

Definition 10 **($\overrightarrow{\mathcal{L}}_\perp$).** *Let $\overrightarrow{\perp}$ be the bottom clause as defined in Definition 2. $\overrightarrow{C}$ is in $\overrightarrow{\mathcal{L}}_\perp$ if and only if there exists a substitution θ such that $\overrightarrow{C}\theta$ is a subsequence of $\overrightarrow{\perp}$.*

Let us consider the examples in the previous section with respect to Definition 10. In Example 1, if we now consider $\overrightarrow{C}$ and $\overrightarrow{\perp}$ as ordered clauses then $\overrightarrow{C} \notin \overrightarrow{\mathcal{L}}_\perp$, because there is no substitution θ such that $\overrightarrow{C}\theta$ can be a subsequence of $\overrightarrow{\perp}$. Similarly in Example 3, $\overrightarrow{C_1'} \notin \overrightarrow{\mathcal{L}}_\perp$, etc.

According to Definition 10, for each ordered clause $\overrightarrow{C}$ in $\overrightarrow{\mathcal{L}}_\perp$ there exists a substitution θ such that $\overrightarrow{C}\theta$ is a subsequence of $\overrightarrow{\perp}$. Thus, there exists a selection function s which maps each literal of $\overrightarrow{C}\theta$ to a literal of $\overrightarrow{\perp}$ and this selection function is strictly increasing. This implies that there is an injective mapping from the literals of $\overrightarrow{C}\theta$ to the literals of $\overrightarrow{\perp}$. Therefore, clause $\overrightarrow{C}$ can be encoded by the substitution θ and a set of integers K, i.e. the range of the selection function s. In Progol's refinement operator, θ and K are maintained for each clause in

[2] Substitution $\theta = \{v_j/u_j\}$ is a variable substitution if all v_j and u_j are variables.

order to decode the clause from $\overrightarrow{\perp}$. Moreover, according to Definition 21, refinements of a clause are constructed by adding literals which are generalisations of a literal from $\overrightarrow{\perp}$. These literals are generated by δ and they all correspond to the same literal l_k from $\overrightarrow{\perp}$. This means that a literal L_i from $\overrightarrow{C}$ is comparable (with respect to Progol's refinement) to a literal M_j from $\overrightarrow{D}$ if L_i and M_j are both mapped to the same literal of $\overrightarrow{\perp}$. This leads to more specific definitions for subsequence and sequential subsumption.

Definition 11 (Subsequence relative to $\perp$). *Let $\overrightarrow{\perp}$ and $\overrightarrow{\mathcal{L}}_\perp$ be as defined in Definition 10 and $\overrightarrow{C} = L_1 \vee L_2 \vee \cdots \vee L_l$ and $\overrightarrow{D} = M_1 \vee M_2 \vee \cdots \vee M_m$ be ordered clauses in $\overrightarrow{\mathcal{L}}_\perp$ such that $\overrightarrow{C} \sqsubseteq^{s_1} \overrightarrow{\perp}$ and $\overrightarrow{D} \sqsubseteq^{s_2} \overrightarrow{\perp}$. $\overrightarrow{C}$ is a subsequence of $\overrightarrow{D}$ relative to $\perp$, denoted by $\overrightarrow{C} \sqsubseteq^{s}_{\perp} \overrightarrow{D}$, if there exists a strictly increasing selection function s such that for each $(i,j) \in s$, $L_i = M_j$ and there exists k, $1 \le k \le n$ and $(i,k) \in s_1$ and $(j,k) \in s_2$.*

In Definition 11 the selection function s maps literal L_i from $\overrightarrow{C}$ to an equivalent literal M_j from $\overrightarrow{D}$ if they both correspond to the same literal from $\overrightarrow{\perp}$.

Definition 12 (Sequential subsumption relative to $\perp$). *Let $\overrightarrow{\perp}$ and $\overrightarrow{\mathcal{L}}_\perp$ be as defined in Definition 10 and $\overrightarrow{C}$ and $\overrightarrow{D}$ be ordered clauses in $\overrightarrow{\mathcal{L}}_\perp$. We say $\overrightarrow{C}$ is a sequential generalization of $\overrightarrow{D}$ relative to $\perp$, denoted by $\overrightarrow{C} \succeq_\perp \overrightarrow{D}$, if there exists a substitution θ such that $\overrightarrow{C}\theta$ is a subsequence of $\overrightarrow{D}$ relative to $\perp$.*

As shown in Example 1, Progol's refinement cannot be weakly complete for $\langle \mathcal{L}, \succeq \rangle$, however, it can be weakly complete for $\langle \overrightarrow{\mathcal{L}}_\perp, \succeq_\perp \rangle$.

Theorem 2. *ρ_0 is weakly complete for $\langle \overrightarrow{\mathcal{L}}_\perp, \succeq_\perp \rangle$.*

A sketch proof for this theorem is given in the Appendix.

5 Ideal Refinement Operators for Sequential Subsumption Order Relative to $\perp$

In this section we define a refinement operator ρ_1 and show that ρ_1 is ideal for $\langle \overrightarrow{\mathcal{L}}_\perp, \succeq_\perp \rangle$. First we define a mapping function which is used in the refinement operator. As mentioned in the previous section, in Progol's refinement operator, a substitution θ and a set of integers K are maintained for each clause in order to decode the clause from $\overrightarrow{\perp}$. In this setting, substitution θ maps variables from $\overrightarrow{C}$ to the variables of $\overrightarrow{\perp}$. The decoding, therefore, requires inverse substitution θ^{-1}. This can be achieved by maintaining the position of variables when the substitution θ is constructed [8]. However, in the mapping function used in this section, substitution θ maps variables from $\overrightarrow{\top}$ to the variables of $\overrightarrow{C}$, where $\overrightarrow{\top}$ is $\overrightarrow{\perp}$ with all variables replaced with new and distinct variables.

Definition 13 (Mapping function c). *Let $\overrightarrow{\bot}$ and $\overrightarrow{\mathcal{L}}_\bot$ be as defined in Definition 10, n be the number of literals in $\bot$. $\overrightarrow{\top}$ is $\overrightarrow{\bot}$ with all variables replaced with new and distinct variables. $\theta_\top$ is a variable substitution such that $\overrightarrow{\top}\theta_\top = \overrightarrow{\bot}$. Let θ be a variable substitution in Θ, where $\Theta = \{\theta' | \theta' \subseteq \hat{\theta}_\top$ and if $\{x/z, z/y\} \subseteq \theta'$ then $x/y \in \theta'\}$ and $\hat{\theta}_\top = \{y/x | \{x/z, y/z\} \subseteq \theta_\top\}$. Let $\mathcal{K}$ be power-set of $\{1, \ldots, n\}$. The mapping function $c : \mathcal{K} \times \Theta \rightarrow \overrightarrow{\mathcal{L}}_\bot$ is defined as follows:*

$$c(\langle K, \theta \rangle) = (\bigvee_{i=1}^{n} l_i | i \in K \text{ and } l_i \text{ is the ith literal of } \overrightarrow{\top})\theta.$$

The mapping function c, maps a tuple $\langle K, \theta \rangle$ into an ordered clause $\overrightarrow{C}$ in $\overrightarrow{\mathcal{L}}_\bot$. This mapping function also makes sure that the literals in $\overrightarrow{C}$ follow the same order as literals in $\overrightarrow{\bot}$. This condition is required for the refinement operator ρ_1 which is intended to be complete for $\langle \overrightarrow{\mathcal{L}}_\bot, \succeq_\bot \rangle$.

The refinement operator ρ_1 is based on Laird's refinement operator [5] adopted for sequential subsumption and a refinement space bounded below by a bottom clause.

Definition 14 (ρ_1). *Let $\overrightarrow{\bot}$ and $\overrightarrow{\mathcal{L}}_\bot$ be as defined in Definition 10, $\overrightarrow{C}$ be an ordered clause in $\overrightarrow{\mathcal{L}}_\bot$, n be the number of literals in $\bot$, k be a natural number, $1 \leq k \leq n$, $\overrightarrow{\top}$, Θ and $\mathcal{K}$ be defined as in Definition 13. Let $K \in \mathcal{K}$, $\theta \in \Theta$, $\overrightarrow{C} = c(\langle K, \theta \rangle)$ and the mapping function c be defined as in Definition 13. $\langle \overrightarrow{C'}, K', \theta' \rangle$ is in $\rho_1(\langle \overrightarrow{C}, K, \theta \rangle)$ if and only if $\overrightarrow{C'} = c(\langle K', \theta' \rangle)$ and either*

1. *$K' = K \cup \{k\}$, $k \notin K$ and $\theta' = \theta$ or*
2. *$K' = K$, $\theta' = \theta\{y'/x'\}$ and $\{y'/x'\} \in \Theta$ where x' and y' are distinct variables in the k_1th and k_2th literals of $\overrightarrow{\top}$ respectively and k_1th and k_2th are in K'.*

In Definition 14, ρ_1 adds a most general literal from $\overrightarrow{\top}$ which has not been added before (item 1) or it applies an elementary variable substitution such that the clause subsumes $\overrightarrow{\bot}$ (item 2). We show that ρ_1 is ideal for $\langle \overrightarrow{\mathcal{L}}_\bot, \succeq_\bot \rangle$. The completeness proof below is similar to the completeness proof for Laird's refinement operator [8,13] adopted for subsumption order relative to $\bot$.

Lemma 1. *Let $\overrightarrow{C}$, $\overrightarrow{D}$ be two ordered clauses in $\overrightarrow{\mathcal{L}}_\bot$ such that $\overrightarrow{C}\theta = \overrightarrow{D}$ for some substitution θ. Then, there exists a ρ_1-chain from $\overrightarrow{C}$ to $\overrightarrow{D}$.*

Proof. Suppose $\overrightarrow{C}$, $\overrightarrow{D}$ are ordered clauses and $\overrightarrow{C}\theta = \overrightarrow{D}$. Then according to Definition 8, $\overrightarrow{C}$ and $\overrightarrow{D}$ have the same predicate symbols at the same positions and therefore can be regarded as atoms. We need to show that there exists a ρ_1-chain from $\overrightarrow{C}$ to $\overrightarrow{D}$ by repeatedly selecting step 2 in Definition 14. The proof is then similar to the proof of this lemma for atoms (Theorem 4 in [10]). □

Lemma 2. *Let $\overrightarrow{C}$, $\overrightarrow{D}$ be two ordered clauses in $\overrightarrow{\mathcal{L}}_\bot$ such that $\overrightarrow{C}$ is a subsequence of $\overrightarrow{D}$ relative to $\bot$. Then, there exists a ρ_1-chain from $\overrightarrow{C}$ to $\overrightarrow{D}$.*

Proof. The proof is by induction on i the number of literals in $\overrightarrow{D}$ but not in $\overrightarrow{C}$. If $i = 0$ then $\overrightarrow{C} = \overrightarrow{D}$, and the empty chain satisfies the lemma. Assume for some j, $0 \leq j < i$, the lemma is true. This implies that there is a ρ_1-chain from $\overrightarrow{C}$ to $\overrightarrow{C}_j$ such that $\overrightarrow{C}_j$ is $\overrightarrow{C}$ with j literals inserted such that $\overrightarrow{C}_j$ is a subsequence of $\overrightarrow{D}$ relative to $\perp$. We show that there is a ρ_1-chain from $\overrightarrow{C}$ to $\overrightarrow{C}_{j+1}$. Let l be the leftmost literal in $\overrightarrow{D}$ which is not in $\overrightarrow{C}_j$. Given that $\overrightarrow{D} \in \overrightarrow{\mathcal{L}}_\perp$ we can assume that l is mapped to the k-th literal of $\perp$. We consider the following two cases: (a) if l is a most general literal with respect to $\overrightarrow{C}_j$, then l is the k-th literal of $\overrightarrow{\top}$ and using item 1 in the definition of ρ_1, $\langle \overrightarrow{C}_{j+1}, K', \theta \rangle \in \rho_1(\langle \overrightarrow{C}_j, K, \theta \rangle)$, where $K' = K \cup \{k\}$. (b) otherwise there is a most general literal l' such that $l'\theta' = l$. In this case, first using item 1 in the definition of ρ_1, $\langle \overrightarrow{C'}_{j+1}, K', \theta \rangle \in \rho_1(\langle \overrightarrow{C}_j, K, \theta \rangle)$ and then according to Lemma 1 (and using item 2 in the definition of ρ_1), $\langle \overrightarrow{C}_{j+1}, K', \theta'' \rangle \in \rho_1^*(\langle \overrightarrow{C'}_{j+1}, K', \theta \rangle)$, where $K' = K \cup \{k\}$ and $\theta'' = \theta\theta'$. Thus, in both cases (a) and (b), there exists a ρ_1-chain from $\overrightarrow{C}$ to $\overrightarrow{C}_{j+1}$ and this completes the proof. □

Theorem 3. *ρ_1 is complete for $\langle \overrightarrow{\mathcal{L}}_\perp, \succeq_\perp \rangle$.*

Proof. Let $\overrightarrow{C}$, $\overrightarrow{D}$ be two ordered clauses in $\overrightarrow{\mathcal{L}}_\perp$ such that, for some θ, $\overrightarrow{C}\theta$ is a subsequence of $\overrightarrow{D}$ relative to $\perp$. If we define $\overrightarrow{E} = \overrightarrow{C}\theta$ then $\overrightarrow{E}$ and $\overrightarrow{C}$ satisfy Lemma 1, hence there is a ρ_1-chain from $\overrightarrow{C}$ to $\overrightarrow{E}$. $\overrightarrow{E}$ is a subsequence of $\overrightarrow{D}$ relative to $\perp$ and according to Lemma 2, there is a ρ_1-chain from $\overrightarrow{E}$ to $\overrightarrow{D}$. Thus, there is a ρ_1-chain from $\overrightarrow{C}$ to $\overrightarrow{D}$ via $\overrightarrow{E}$. □

According to Definition 14, the refinement operator ρ_1 works on an encoding of a clause, i.e. $\langle K, \theta \rangle$ rather than the clause itself. In the following we define the order relation for the encoding tuples $\langle K, \theta \rangle$, used in the mapping function c. Then, we show that the mapping function c is order-embedding.

Definition 15. *Let $\mathcal{K}$ and Θ be defined as in Definition 13 and $K_1, K_2 \in \mathcal{K}$ and $\theta_1, \theta_2 \in \Theta$. $\langle K_1, \theta_1 \rangle \subseteq \langle K_2, \theta_2 \rangle$ if and only if $K_1 \subseteq K_2$ and $\theta_1 \subseteq \theta_2$. $\theta_1 \subseteq \theta_2$ if and only if there exists a substitution θ such that $\theta_2 = \theta_1\theta$. $\langle K_1, \theta_1 \rangle \sim \langle K_2, \theta_2 \rangle$ if and only if $\langle K_1, \theta_1 \rangle \subseteq \langle K_2, \theta_2 \rangle$ and $\langle K_2, \theta_2 \rangle \subseteq \langle K_1, \theta_1 \rangle$.*

Theorem 4. *Let $\mathcal{K}$ and Θ and mapping function c be defined as in Definition 13 and $K_1, K_2 \in \mathcal{K}$ and $\theta_1, \theta_2 \in \Theta$. $c(\langle K_1, \theta_1 \rangle) \succeq_\perp c(\langle K_2, \theta_2 \rangle)$ if and only if $\langle K_1, \theta_1 \rangle \subseteq \langle K_2, \theta_2 \rangle$.*

Proof. $\Rightarrow$: Let $\overrightarrow{C}$, $\overrightarrow{D}$ be ordered clauses such that $\overrightarrow{C} = c(\langle K_1, \theta_1 \rangle)$ and $\overrightarrow{D} = c(\langle K_2, \theta_2 \rangle)$. Assume $\overrightarrow{C} \succeq_\perp \overrightarrow{D}$, then according to Theorem 3 there is a ρ_1-chain from $\overrightarrow{C}$ to $\overrightarrow{D}$. Let this ρ_1-chain be $C = C'_0 \succeq_\perp C'_1 \succeq_\perp \cdots \succeq_\perp C'_m = D$ where $\langle \overrightarrow{C'}_{i+1}, K'_{i+1}, \theta'_{i+1} \rangle \in \rho_1(\langle \overrightarrow{C'}_i, K'_i, \theta'_i \rangle)$, $0 \leq i < m$. According to the definition of ρ_1, in each refinement step either 1) $K'_i \subseteq K'_{i+1}$ and $\theta'_{i+1} = \theta'_i$ or 2) $K'_{i+1} = K'_i$ and $\theta'_i \subseteq \theta'_{i+1}$. Then it is always the case that $K'_i \subseteq K'_{i+1}$ and $\theta'_i \subseteq \theta'_{i+1}$, where $K'_0 = K_1, K'_m = K_2, \theta'_0 = \theta_1, \theta'_m = \theta_2$. Thus, $K_1 \subseteq K_2$ and $\theta_1 \subseteq \theta_2$.

$\Leftarrow$: Let $\overrightarrow{C} = c(K_1, \theta_1) = (\bigvee l_i | i \in K_1)\theta_1$ and $\overrightarrow{D} = c(K_2, \theta_2) = (\bigvee l_j | j \in K_2)\theta_2$ such that $K_1 \subseteq K_2$ and $\theta_1 \subseteq \theta_2$. According to Definition 15, $\theta_2 = \theta_1\theta$ for some substitution θ. Then, given $K_1 \subseteq K_2$, for every literal $l_i\theta_1$ from $\overrightarrow{C}\theta$, we have a literal $l_i\theta_1\theta$ from $\overrightarrow{D}$ where $l_i\theta_1$ and $l_i\theta_1\theta$ are both mapped to the same literal l_i from $\overrightarrow{\top}$ (Definition 13). Thus, $\overrightarrow{C}\theta$ is a subsequence of $\overrightarrow{D}$ relative to $\perp$ and therefore $\overrightarrow{C} \succeq_\perp \overrightarrow{D}$. □

In the following we show the properness and the idealness of ρ_1 for $\langle \overrightarrow{\mathcal{L}}_\perp, \succeq_\perp \rangle$.

Lemma 3. *Let $\overrightarrow{C}$ and $\overrightarrow{D}$ be ordered clauses. $\overrightarrow{C} \sim_\perp \overrightarrow{D}$ if and only if $\overrightarrow{C}$ and $\overrightarrow{D}$ are alphabetical variants.*

Proof. $\Rightarrow$: Suppose $\overrightarrow{C} \sim_\perp \overrightarrow{D}$, then we have $\overrightarrow{C} \succeq_\perp \overrightarrow{D}$ and $\overrightarrow{D} \succeq_\perp \overrightarrow{C}$. Thus, there are substitutions θ_1 and θ_2 such that $\overrightarrow{C}\theta_1$ is a subsequence of $\overrightarrow{D}$ relative to $\perp$ and $\overrightarrow{D}\theta_2$ is a subsequence of $\overrightarrow{C}$ relative to $\perp$. Let $\overrightarrow{C} = L_1 \vee L_2 \vee \cdots \vee L_l$ and $\overrightarrow{D} = M_1 \vee M_2 \vee \cdots \vee M_m$. Therefore, there are strictly increasing selection functions s_1 and s_2 such that for each $(i, j) \in s_1$, $L_i\theta_1 = M_j$ and for each $(i, j) \in s_2$, $M_i\theta_2 = L_j$. Given that s_1 and s_2 are strictly increasing functions, there is a one-to-one mapping between literals of $\overrightarrow{C}$ and $\overrightarrow{D}$ such that $m = n$, $L_i\theta_1 = M_i$ and $M_i\theta_2 = L_i$. Therefore it holds that $\overrightarrow{C}\theta_1 = \overrightarrow{D}$ and $\overrightarrow{D}\theta_2 = \overrightarrow{C}$. Hence, $\overrightarrow{C}$ and $\overrightarrow{D}$ are alphabetical variants.

$\Leftarrow$: Suppose $\overrightarrow{C}$ and $\overrightarrow{D}$ are alphabetical variants. Therefore there are substitutions θ_1 and θ_2 such $\overrightarrow{C}\theta_1 = \overrightarrow{D}$ and $\overrightarrow{D}\theta_2 = \overrightarrow{C}$. Then it follows from Definition 12 that $\overrightarrow{C} \succeq_\perp \overrightarrow{D}$ and $\overrightarrow{D} \succeq_\perp \overrightarrow{C}$ and therefore $\overrightarrow{C} \sim_\perp \overrightarrow{D}$. □

Lemma 4. *Let $\mathcal{K}$ and Θ and mapping function c be defined as in Definition 13 and $K, \{k\} \in \mathcal{K}$ such that $k \notin K$ and $\theta \in \Theta$. Then, $c(\langle K \cup \{k\}, \theta \rangle) \succ_\perp c(K, \theta)$.*

Proof. Suppose $c(\langle K \cup \{k\}, \theta \rangle) \not\succ_\perp c(K, \theta)$. We know from Theorem 4 that $c(\langle K \cup \{k\}, \theta \rangle) \succeq_\perp c(K, \theta)$, and therefore $c(\langle K \cup \{k\}, \theta \rangle) \sim_\perp c(K, \theta)$. According to Lemma 3, $c(\langle K \cup \{k\}, \theta \rangle)$ and $c(K, \theta)$ must be alphabetical variants, contradicting $k \notin K$. Thus, $c(\langle K \cup \{k\}, \theta \rangle) \succ_\perp c(K, \theta)$. □

Lemma 5. *Let $\mathcal{K}$ and Θ, $\top$ and mapping function c be defined as in Definition 13 and $K \in \mathcal{K}$, $\{y/x\}, \theta \in \Theta$ where x and y are distinct variables in the k_1th and k_2th literals of $\overrightarrow{\top}$ respectively and k_1th and k_2th are in K. Then, $c(\langle K, \theta\{y/x\} \rangle) \succ_\perp c(K, \theta)$.*

Proof. Suppose $c(\langle K, \theta\{y/x\} \rangle) \not\succ_\perp c(K, \theta)$. We know from Theorem 4 that $c(\langle K, \theta\{y/x\} \rangle) \succeq_\perp c(K, \theta)$, and therefore $c(\langle K, \theta\{y/x\} \rangle) \sim_\perp c(K, \theta)$. According to Lemma 3, $c(\langle K, \theta\{y/x\} \rangle)$ and $c(K, \theta)$ must be alphabetical variants. Thus, $\{y/x\}$ must be a renaming subsumption, i.e. x is either equal to y or it does not occur in $c(K, \theta)$, contradicting the assumption. Thus, $c(\langle K, \theta\{y/x\} \rangle) \succ_\perp c(K, \theta)$. □

Theorem 5. *ρ_1 is proper for $\langle \overrightarrow{\mathcal{L}}_\perp, \succeq_\perp \rangle$.*

Proof. If $\langle C', \theta', K'\rangle \in \rho_1(\langle C, \theta, K\rangle)$ is generated by item 1 in the definition of ρ_1, then $\overrightarrow{C} \succ_\perp \overrightarrow{D}$ follows from Lemma 4. If it is generated by item 2 in the definition of ρ_1, then $\overrightarrow{C} \succ_\perp \overrightarrow{D}$ follows from Lemma 5. □

Theorem 6. *ρ_1 is ideal for $\langle \overrightarrow{\mathcal{L}}_\perp, \succeq_\perp\rangle$.*

Proof. Locally finiteness follows from the definition of ρ_1 and the fact that there are finite number of literals and variables in $\perp$. Completeness and properness were proved in Theorem 3 and Theorem 5 respectively. □

In the following we study the morphism between $\langle \overrightarrow{\mathcal{L}}_\perp, \succeq_\perp\rangle$ and $\langle \mathcal{K} \times \Theta, \subseteq\rangle$. According to Theorem 4, the mapping function c is an order-embedding. The following theorem shows that c is also an order-isomorphism.

Theorem 7. *The mapping function $c : \mathcal{K} \times \Theta \rightarrow \overrightarrow{\mathcal{L}}_\perp$ as defined in Definition 13 is an order-isomorphism.*

Proof. According to Theorem 4, the mapping function c is an order-embedding, so we only need to prove that c is onto. Let $\overrightarrow{C}$ be an ordered clause in $\overrightarrow{\mathcal{L}}_\perp$, then according to Definition 10, there exist substitution θ and selection function s such that $\overrightarrow{C}\theta \sqsubseteq^s \overrightarrow{\perp}$. From Definition 13 we have $\overrightarrow{\top}\theta_\top = \overrightarrow{\perp}$ and therefore $\overrightarrow{C}\theta \sqsubseteq^s \overrightarrow{\top}\theta_\top$ and this implies $\overrightarrow{C} \sqsubseteq^s \top\theta_\top\theta^{-1}$. Thus, $\overrightarrow{C}$ can be defined as $\overrightarrow{C} = c(K, \theta') = (\bigvee l_i | i \in K)\theta'$, where $\theta' = \theta_\top\theta^{-1}$ and K is the range of the selection function s. □

The proposition below follows directly from Theorem 7.

Proposition 1. *Let $\mathcal{K}$ and Θ and mapping function c be defined as in Definition 13 and $K_1, K_2 \in \mathcal{K}$ and $\theta_1, \theta_2 \in \Theta$. $c(K, \theta) \sim_\perp c(K', \theta')$ if and only if $\langle K, \theta\rangle \sim \langle K', \theta'\rangle$.*

It is known that if a mapping is order-isomorphism it is also a lattice isomorphism.

Theorem 8 ([3]). *Let $\langle L, \wedge, \vee\rangle$ and $\langle K, \cap, \cup\rangle$ be lattices and $f : L \rightarrow K$. f is order-isomorphism if and only if it is a lattice isomorphism.*

According to this theorem the mapping c is a lattice isomorphism. Thus, it can be shown that $\langle \overrightarrow{\mathcal{L}}_\perp, \succeq_\perp\rangle$ and $\langle \mathcal{K} \times \Theta, \subseteq\rangle$ are two isomorphic lattices. This also means that the mapping c is a lattice homomorphism. The proposition below follows from c being a homomorphism.

Proposition 2. *Let $\mathcal{K}$ and Θ and mapping function c be defined as in Definition 13 and $K_1, K_2 \in \mathcal{K}$ and $\theta_1, \theta_2 \in \Theta$. Mapping c is joint-preserving and meet-preserving that is:*

1. $lgg_\perp(c(\langle K_1, \theta_1\rangle, c(\langle K_2, \theta_2\rangle) = c(\langle K_1 \cap K_2, \theta_1 \cap \theta_2\rangle)$
2. $mgi_\perp(c(\langle K_1, \theta_1\rangle, c(\langle K_2, \theta_2\rangle) = c(\langle K_1 \cup K_2, \theta_1 \cup \theta_2\rangle)$

According to Proposition 2, the least general generalisation ($lgg_\perp$) and the most specific instance ($mgi_\perp$) for $\langle \overrightarrow{\mathcal{L}}_\perp, \succeq_\perp \rangle$ can be defined based on the the joint and the meet operations for $\langle \mathcal{K} \times \Theta, \subseteq \rangle$. Note that if two lattices are isomorphic then for practical purposes they are identical and differ only in the notation of their elements. The morphism between $\langle \overrightarrow{\mathcal{L}}_\perp, lgg_\perp, mgi_\perp \rangle$ and $\langle \mathcal{K} \times \Theta, \cap, \cup \rangle$ is important from a practical point of view. For example, the least general generalisation (lgg) of clauses, introduced by Plotkin [9], is an important operator for ILP. However, in the general subsumption order the construction of lgg can be inefficient (e.g. the cardinality of the lgg of two clauses is bounded by the product of the cardinalities of the two clauses). On the other hand, efficient operators can be implemented for least generalisation and greatest specialisation in the sequential subsumption order relative to a bottom clause.

6 Subsumption Order Relative to $\perp$

The purpose of the previous sections was to characterise Progol's refinement and the subsumption sub-lattice which is searched by Progol. We defined sequential subsumption order relative to $\perp$ and studied the properties of this special case of subsumption. In this section we show how some limitations of Progol's refinement operator can be addressed by relaxing conditions of sequential subsumption. In particular we address the first type of incompleteness, which is believed to be more problematic than the second type of incompleteness[1]. In this section we define a refinement operator which is less restricted than ρ_1. As demonstrated in section 3, the first type of Progol's refinement incompleteness is due to the choice of ordering of literals in $\perp$ and the fact that clauses are considered as subsequences of $\perp$. This condition was embedded in the definitions of the sequential subsumption and the refinement operator ρ_1. However, more relaxed conditions can be defined for subsumption and refinement operators relative to $\perp$. Note that in the previous definitions and theorems we only needed to assume that the selection functions are injective so that we can encode every literal of a clause by a k index from $\perp$. Therefore a less restricted ordering can be defined by using a selection function which is injective rather than strictly increasing.

Definition 16 (Ordered subset). *Let $\overrightarrow{C} = L_1 \vee L_2 \vee \cdots \vee L_l$ and $\overrightarrow{D} = M_1 \vee M_2 \vee \cdots \vee M_m$ be ordered clauses. $\overrightarrow{C}$ is an ordered subset of $\overrightarrow{D}$, denoted by $\overrightarrow{C} \subseteq^s \overrightarrow{D}$, if there exists an injective selection function s such that for each $(i, j) \in s$, $L_i = M_j$.*

By choosing s to be an injective function, we make sure that clauses can still be encoded by a set of k indexes. However, these clauses do not need to follow the same order as literals in $\perp$. In the following, we give new definitions for the mapping function and the refinement operator for this less restricted subsumption order.

Definition 17 ($\mathcal{L}_\perp$). *Let $\overrightarrow{\perp}$ be the bottom clause as defined in Definition 2. $\overrightarrow{C}$ is in $\mathcal{L}_\perp$ if and only if there exists a substitution θ such that $\overrightarrow{C}\theta$ is an ordered subset of $\overrightarrow{\perp}$.*

Definition 18 (Mapping function c'). *Let $\overrightarrow{\perp}$ and $\mathcal{L}_\perp$ be as defined in Definition 17, n be the number of literals in $\perp$. Let $\overrightarrow{\top}$, $\theta_\top$, θ, Θ, $\mathcal{K}$ be as defined in Definition 13. The mapping function $c' : \mathcal{K} \times \Theta \to \mathcal{L}_\perp$ is defined as follows:*

$$c'(\langle K, \theta\rangle) = (\bigvee_{i \in K} l_i | l_i \textit{ is the ith literal of } \overrightarrow{\top})\theta.$$

In the definition of c', unlike in c, literals l_i do not need to follow the same order as literals in $\overrightarrow{\top}$. In the following we define a refinement operator, ρ_2, which is similar to ρ_1 but uses the mapping function c' instead of c.

Definition 19 (ρ_2). *Let $\overrightarrow{\perp}$ and $\mathcal{L}_\perp$ be as defined in Definition 17, $\overrightarrow{C}$ be an ordered clause in $\mathcal{L}_\perp$, n be the number of literals in $\perp$, k be a natural number, $1 \leq k \leq n$, $\overrightarrow{\top}$, Θ and $\mathcal{K}$ be defined as in Definition 18. Let $K \in \mathcal{K}$, $\theta \in \Theta$, $\overrightarrow{C} = c'(\langle K, \theta\rangle)$ and the mapping function c' be defined as in Definition 18. $\langle \overrightarrow{C'}, K', \theta'\rangle$ is in $\rho_2(\langle \overrightarrow{C}, K, \theta\rangle)$ if and only if $\overrightarrow{C'} = c'(\langle K', \theta'\rangle)$ and either*

1. *$K' = K \cup \{k\}$, $k \notin K$ and $\theta' = \theta$ or*
2. *$K' = K$, $\theta' = \theta\{y'/x'\}$ and $\{y'/x'\} \in \Theta$ where x' and y' are distinct variables in the k_1th and k_2th literals of $\overrightarrow{\top}$ respectively and k_1th and k_2th are in K'.*

The following example demonstrates how the first type of incompleteness (in Example 1) is addressed in ρ_2.

Example 7. Let $\overrightarrow{C}$ and $\overrightarrow{\perp}$ be as defined in Example 1. Progol's refinement cannot generate C (i.e. $C \notin \rho^*(\square))$) and also $\langle \overrightarrow{C}, K, \theta\rangle \notin \rho_1^*(\langle \square, \emptyset, \emptyset\rangle)$. However, Table 1 shows that $\langle \overrightarrow{C}, K, \theta\rangle \in \rho_2^*(\langle \square, \emptyset, \emptyset\rangle)$, where $K = \{1, 2, 5, 3\}$ and $\theta = \{V_4/V_1, V_{12}/V_5, V_{13}/V_2, V_6/V_{14}, V_7/V_2\}$ and $\overrightarrow{\top}$ is the clause:

$$\begin{aligned}\text{mult}(V_1, V_2, V_3) \leftarrow\ & \text{dec}(V_4, V_5), \text{plus}(V_6, V_7, V_8), \text{plus}(V_9, V_{10}, V_{11}),\\ & \text{mult}(V_{12}, V_{13}, V_{14}), \text{mult}(V_{15}, V_{16}, V_{17}).\end{aligned} \qquad \diamond$$

This example shows that ρ_2 can address the incompleteness of ρ demonstrated in Example 1. However, ρ_2 is also more redundant than ρ (e.g. different permutations of the same clause could be generated). On the other hand, as mentioned in Section 3, a refinement operator cannot be both complete and non-redundant. Given that in the new definitions the selection functions are injective, we can encode every literal of a clause by a k index from $\perp$. Therefore, the properties mentioned in Section 5 for the mapping function c also hold for c'. By different conditions on the selection functions in Definition 16 we can get different kind of subsumption orders. For example, if the selection function is monotonically increasing then we will have a subsumption order which allows each literal of $\perp$

Table 1. Application of ρ_2 in Example 7

C'	θ'	K'
$\square$	$\emptyset$	$\emptyset$
$mult(V_1, V_2, V_3) \leftarrow$	$\emptyset$	$\{1\}$
$mult(V_1, V_2, V_3) \leftarrow dec(V_4, V_5)$	$\emptyset$	$\{1, 2\}$
$mult(V_1, V_2, V_3) \leftarrow dec(V_1, V_5)$	$\{V_4/V_1\}$	$\{1, 2\}$
$mult(V_1, V_2, V_3) \leftarrow dec(V_1, V_5), mult(V_{12}, V_{13}, V_{14})$	$\{V_4/V_1\}$	$\{1, 2, 5\}$
$mult(V_1, V_2, V_3) \leftarrow dec(V_1, V_5), mult(V_5, V_{13}, V_{14})$	$\{V_4/V_1, V_{12}/V_5\}$	$\{1, 2, 5\}$
$mult(V_1, V_2, V_3) \leftarrow dec(V_1, V_5), mult(V_5, V_2, V_{14})$	$\{V_4/V_1, V_{12}/V_5, V_{13}/V_2\}$	$\{1, 2, 5\}$
$mult(V_1, V_2, V_3) \leftarrow dec(V_1, V_5), mult(V_5, V_2, V_{14}),$ $plus(V_6, V_7, V_8)$	$\{V_4/V_1, V_{12}/V_5, V_{13}/V_2\}$	$\{1, 2, 5, 3\}$
$mult(V_1, V_2, V_3) \leftarrow dec(V_1, V_5), mult(V_5, V_2, V_{14}),$ $plus(V_{14}, V_7, V_8)$	$\{V_4/V_1, V_{12}/V_5, V_{13}/V_2,$ $V_6/V_{14}\}$	$\{1, 2, 5, 3\}$
$mult(V_1, V_2, V_3) \leftarrow dec(V_1, V_5), mult(V_5, V_2, V_{14}),$ $plus(V_{14}, V_2, V_8)$	$\{V_4/V_1, V_{12}/V_5, V_{13}/V_2,$ $V_6/V_{14}, V_7/V_2\}$	$\{1, 2, 5, 3\}$

to be selected more than once[3]. This will address the second type of Progol's incompleteness mentioned before. However, the selection functions are not injective and therefore the encoding and the morphism we described in this paper are not applicable.

7 Related Work and Discussion

Progol's refinement operator and its incompleteness with respect to the general subsumption order were initially discussed in [7]. The purpose of the present paper was to characterising Progol's refinement space and to give an analysis of refinement operators for this space. In a previous attempt, the authors of [1] suggested weak subsumption for characterising Progol's refinement space. However, as we have shown in this paper, weak subsumption cannot capture all aspects of Progol's refinement. They also considered refinement operators which, as the operators considered in this paper, are based on Laird's operator. In this paper we used an encoding of clauses with respect to a bottom clause. In this encoding each clause is represented by a tuple $\langle K, \theta \rangle$ and it can be constructed from $\overrightarrow{\top}$ as described in Definition 13. This idea was first used in [12] where the substitution θ is encoded as a binding matrix which maps the variables of $\overrightarrow{\top}$ to the variables of a clause with respect to the bottom clause. The morphism between the lattice of variable bindings and the subsumption lattice was also studied in [12]. A subsumption relation for ordered clauses (i.e. ordered subsumption) is studied in [4]. It is shown that, in the defined subsumption, the least generalisation of two ordered clauses does not exist and that the subsumption testing for ordered clauses is NP-complete. The subsequence relation considered in [4], assumes a mapping function which is monotonically increasing (rather than strictly increasing). As mentioned in the previous section, this leads to a different subsumption from the one considered in this paper (i.e. sequential subsumption) and the results from this paper are not applicable.

[3] In this case, Definition 16 will be identical to the definition of subsequence considered in [4].

8 Conclusions

In this paper we have studied refinement operators for a hypotheses space bounded by a most specific (bottom) clause. We introduced a subsumption order relative to a bottom clause and demonstrated how Progol's refinement can be characterised with respect to this order. We also proved that ideal refinement operators exist for this order. It was shown that efficient operators can be defined for least generalisation and greatest specialisation in the subsumption order relative to a bottom clause. The theoretical results presented in this paper can be applied to ILP systems which use Inverse Entailment (IE) as well as any other ILP system which uses a bottom clause to restrict the search space.

Acknowledgments

We would like to thank the three anonymous reviewers for their comments. The first author was supported by the BBSRC grant BB/C519670/1. The second author would like to thank the Royal Academy of Engineering and Microsoft for funding his present 5 year Research Chair.

References

1. Badea, L., Stanciu, M.: Refinement operators can be (weakly) perfect. In: Džeroski, S., Flach, P.A. (eds.) ILP 1999. LNCS (LNAI), vol. 1634, pp. 21–32. Springer, Heidelberg (1999)
2. Chang, C., Lee, R.: Symbolic Logic and Mechanical Theorem Proving. Academic Press, London (1973)
3. Davey, B.A., Priestley, H.A.: Introduction to Lattices and Order. Cambridge University Press, Cambridge (2002)
4. Kuwabara, M., Ogawa, T., Hirata, K., Harao, M.: On generalization and subsumption for ordered clauses. In: Proceedings of the JSAI 2005 Workshops, pp. 212–223 (2006)
5. Laird, P.D.: Learning from good data and bad. PhD thesis, Yale University (1987)
6. Lee, S.D., De Raedt, L.: Constraint Based Mining of First Order Sequences in SeqLog. Database Support for Data Mining Applications, 155–176 (2003)
7. Muggleton, S.: Inverse entailment and Progol. New Generation Computing 13, 245–286 (1995)
8. Nienhuys-Cheng, S.-H., de Wolf, R.: Foundations of Inductive Logic Programming. LNCS (LNAI), vol. 1228. Springer, Heidelberg (1997)
9. Plotkin, G.D.: Automatic Methods of Inductive Inference. PhD thesis, Edinburgh University (August 1971)
10. Reynolds, J.C.: Transformational systems and the algebraic structure of atomic formulas. In: Meltzer, B., Michie, D. (eds.) Machine Intelligence 5, pp. 135–151. Edinburgh University Press, Edinburgh (1969)
11. Srinivasan, A.: The Aleph Manual. University of Oxford, Oxford (2007)

12. Tamaddoni-Nezhad, A., Muggleton, S.H.: Searching the subsumption lattice by a genetic algorithm. In: Cussens, J., Frisch, A.M. (eds.) ILP 2000. LNCS (LNAI), vol. 1866, pp. 243–252. Springer, Heidelberg (2000)
13. van der Laag, P.: An Analysis of Refinement Operators in Inductive Logic Programming. Tinbergen Institute Research Series, Rotterdam (1995)

A Progol's Refinement Operator ρ_0

The following definition describes a bottom clause $\bot_i$ for a depth-bounded mode language $\mathcal{L}_i(M)$ as defined in [7]. In this paper, we refer to $\bot_i$ and $\mathcal{L}_i(M)$ as $\bot$ and $\mathcal{L}$ respectively.

Definition 20. Most-specific clause $\bot_i$. *Let h, i be natural numbers B be a set of Horn clauses, $e = a \leftarrow b_1, .., b_n$ be a definite clause, M be a set of mode declarations containing exactly one modeh m such that $a(m) \succeq a$ and $\bot$ be the most-specific (potentially infinite) definite clause such that $B \wedge \bot \wedge \overline{e} \vdash_h \Box$. $\bot_i$ is the most-specific clause in $\mathcal{L}_i(M)$ such that $\bot_i \succeq \bot$.*

The refinement operator ρ defined in [7] allows more than one literal in a clause to be mapped to the same literal in $\bot$. However, in Progol's implementation of the refinement operator, index k is incremented after each step for the sake of efficiency. This means each literal of $\bot$ can be considered only once. In the following, we give a revised definition (ρ_0) which describes the refinement operator as implemented in Progol. This also includes a revised definition for function δ.

Definition 21. Progol refinement operator ρ_0. *Let h, i, B, e, M and $\bot_i$ be defined as in Definition 20 and let n be the cardinality of $\bot_i$. Let k be a natural number, $1 \leq k \leq n$. Let C be a clause in $\mathcal{L}_i(M)$ and θ be a substitution such that $C\theta \subseteq \bot_i$. Below a literal l corresponding to a mode m_l in M is denoted simply as $p(v_1, .., v_m)$ despite the sign of m_l and function symbols in $a(m_l)$. A variable is splittable if it corresponds to a +type or -type in a modeh or if it corresponds to a -type in a modeb. $\langle C', \theta', k' \rangle$ is in $\rho_0(\langle C, \theta, k \rangle)$ if and only if either*

1. *$C' = C \vee l$, $k' = k + 1$, $k < n$ and $\langle l, \theta' \rangle$ is in $\delta(\theta, k)$ and $C' \in \mathcal{L}_i(M)$ or*
2. *$C' = C$, $k' = k + 1$, $\theta' = \theta$ and $k < n$.*

$\langle p(v_1, .., v_m), \theta'_m \rangle$ is in $\delta(\theta, k)$ if and only if $l_k = p(u_1, .., u_m)$ is the kth literal of $\bot_i$, $\theta'_0 = \theta$ and θ'_j for each j, $1 \leq j \leq m$ is defined as follows:

1. *if $v_j/u_j \in \theta'_{j-1}$ then $\theta'_j = \theta'_{j-1}$ or*
2. *if u_j is splitable then $\theta'_j = \theta'_{j-1} \cup \{v_j/u_j\}$ where v_j is a new variable not in $dom(\theta'_{j-1})$.*

In Definition 21, the refinement operator ρ_0, as ρ in [7], is defined for clauses in $\mathcal{L}_i(M)$. However, ρ_0 can be also defined for clauses in $\overrightarrow{\mathcal{L}}_\bot$ if we let C and C' to be ordered clauses in $\overrightarrow{\mathcal{L}}_\bot$ and $\overrightarrow{C}\theta$ be a subsequence of the bottom clause (rather than a subset). In this case, it can be shown that ρ_0 is weakly complete for $\langle \overrightarrow{\mathcal{L}}_\perp, \succeq_\perp \rangle$.

Theorem 2. ρ_0 is weakly complete for $\langle \overrightarrow{\mathcal{L}}_\perp, \succeq_\perp \rangle$.

Sketch proof. We need to show that $\rho_0^*(\langle \Box, \emptyset, 1 \rangle) = \overrightarrow{\mathcal{L}}_\perp$. We show that for each $\overrightarrow{C} \in \overrightarrow{\mathcal{L}}_\perp$, there exists a ρ_0-chain from $\Box$ to $\overrightarrow{C'}$ where $\overrightarrow{C'}$ and $\overrightarrow{C}$ are alphabetical variants. The proof is by induction on i, the number of literals in $\overrightarrow{C}$. If $i = 0$ then $\overrightarrow{C} = \Box$, and the empty chain satisfies the theorem. Assume for some j, $0 \leq j < i$, that the lemma is true. This implies that there is a ρ_0-chain from $\Box$ to $\overrightarrow{C}_j$ such that $\overrightarrow{C}_j$ is an ordered clause in $\overrightarrow{\mathcal{L}}_\perp$ with j literals added from $\overrightarrow{C}$. Therefore, there is a substitution θ such that $\overrightarrow{C}_j\theta$ is a subsequence of $\perp$ and we assume that the j-th literal of $\overrightarrow{C}_j$ is mapped to the k-th literal of $\perp$. Let $\overrightarrow{C}_{j+1} = \overrightarrow{C}_j \vee l$, where l is the leftmost literal of $\overrightarrow{C}$ which is not in $\overrightarrow{C}_j$ and l is mapped to the k'-th literal of $\perp$, where $k < k'$ (because $\overrightarrow{C}_j$ and $\overrightarrow{C}_{j+1}$ are sequential generalisations of $\perp$). Then there exists a ρ_0-chain from $\langle \overrightarrow{C}_j, \theta, k \rangle$ to $\langle \overrightarrow{C}_j, \theta, k' \rangle$ by repeatedly selecting item 2 in the definition of ρ_0 in order to skip $k' - k$ literals of $\perp$. According to the definition of δ, there exists $\langle l', \theta' \rangle$ in $\delta(\theta, k')$ such that, by construction, l and l' are variants. Therefore, by selecting item 1 in the definition of ρ_0, $\overrightarrow{C'}_{j+1} = \overrightarrow{C}_j \vee l$ is a variant of $\overrightarrow{C}_{j+1} = \overrightarrow{C}_j \vee l$, where $\langle \overrightarrow{C'}_{j+1}, \theta', k' + 1 \rangle \in \rho_0(\langle \overrightarrow{C}_j, \theta, k' \rangle)$. Thus, there is a ρ_0-chain from $\Box$ to a a variant of $\overrightarrow{C}_{j+1}$ and this completes the proof. $\Box$