Mach Learn (2009) 76: 37–72
DOI 10.1007/s10994-009-5117-7

The lattice structure and refinement operators for the hypothesis space bounded by a bottom clause

Alireza Tamaddoni-Nezhad · Stephen Muggleton

Received: 17 November 2008 / Revised: 6 April 2009 / Accepted: 17 April 2009 /
Published online: 12 June 2009
Springer Science+Business Media, LLC 2009

Abstract Searching the hypothesis space bounded below by a bottom clause is the basis of several state-of-the-art ILP systems (e.g. Progol, Aleph). These systems use refinement operators together with search heuristics to explore a bounded hypothesis space. It is known that the search space of these systems is limited to a sub-graph of the general subsumption lattice. However, the structure and properties of this sub-graph have not been properly characterised. In this paper firstly, we characterise the hypothesis space considered by the ILP systems which use a bottom clause to constrain the search. In particular, we discuss refinement in Progol as a representative of these ILP systems. Secondly, we study the lattice structure of this bounded hypothesis space. Thirdly, we give a new analysis of refinement operators, least generalisation and greatest specialisation in the subsumption order relative to a bottom clause. The results of this study are important for better understanding of the constrained refinement space of ILP systems such as Progol and Aleph, which proved to be successful for solving real-world problems (despite being incomplete with respect to the general subsumption order). Moreover, characterising this refinement sub-lattice can lead to more efficient ILP algorithms and operators for searching this particular sub-lattice. For example, it is shown that, unlike for the general subsumption order, efficient least generalisation operators can be designed for the subsumption order relative to a bottom clause.

Keywords Subsumption lattice · Refinement operators · Ordered clauses · Sequential subsumption · Subsumption relative to a bottom clause

1 Introduction

Searching the hypothesis space bounded below by a bottom clause is the basis of several state-of-the-art ILP systems. In particular ILP systems such as Progol (Muggleton 1995)

Editors: Filip Zelezny and Nada Lavrac

A. Tamaddoni-Nezhad (✉) · S. Muggleton
Department of Computing, Imperial College London, London, UK
e-mail: atn@doc.ic.ac.uk

S. Muggleton
e-mail: shm@doc.ic.ac.uk

and Aleph (Srinivasan 2007), which use Inverse Entailment (IE), are based on clause refinement through the hypothesis space bounded by a most specific clause. These systems use refinement operators together with a search method to explore a bounded hypothesis space. It is known that the search space of these systems is limited to a sub-graph of the general subsumption lattice. However, the structure and properties of this sub-graph have not been properly characterised. In a previous paper (Tamaddoni-Nezhad and Muggleton 2008) we gave an analysis of refinement operators in a Progol-like ILP system. This analysis has been extended in the present paper. In particular we further discuss the lattice structure and the properties of the subsumption order relative to a bottom clause. In this paper firstly, we characterise the hypothesis space considered by the ILP systems which use a bottom clause to constrain the search. In particular we discuss refinement in Progol as a representative of these ILP systems. Secondly, we study the lattice structure of this bounded hypothesis space. Thirdly, we give a new analysis of refinement operators, least generalisation and greatest specialisation in the subsumption order relative to a bottom clause.

The results of this study are important for better understanding of the constrained refinement space of ILP systems such as Progol and Aleph which proved to be successful for solving real-world problems (despite being incomplete with respect to general subsumption order). Moreover, characterising this refinement sub-lattice can lead to more efficient ILP algorithms and operators for searching this particular sub-lattice. For example, it is shown that, unlike for the general subsumption order, efficient least generalisation operators can be designed for the subsumption order relative to a bottom clause. This idea is the basis of a new ILP system (Muggleton et al. 2009). The theoretical results presented in this paper are applicable to ILP systems such as Progol and Aleph which use some form of Inverse Entailment (IE). Moreover, these results are also applicable to other ILP systems which use a bottom clause to restrict the search space. These include ILP systems which use stochastic algorithms to explore the hypothesis space bounded by a bottom clause (e.g. Srinivasan 2000; Tamaddoni-Nezhad and Muggleton 2000; Zelezny et al. 2003; Muggleton and Tamaddoni-Nezhad 2007; Duboc et al. 2008). The search space of these systems can be characterised by a quasi-ordered set which is described in this paper.

This paper is organised as follows. In Sect. 2 we review some of basic concepts from ordered sets, lattices and inductive logic programming which are used in the definitions and theorems in this paper. In Sect. 3 we discuss clause refinement in Progol as a representative of ILP systems which use a bottom clause to constrain the search. Ordered clauses and sequential subsumption are discussed in Sect. 4. In order to characterise refinement in a Progol-like ILP system, Sect. 5 defines subsumption order relative to a bottom clause and describes the properties of this subsumption order. The lattice structure and properties of the subsumption order relative to a bottom clause are examined in Sect. 6. In Sect. 7, refinement operators are defined for subsumption order relative to a bottom clause and the properties of these operators are discussed. Section 8 describes alternative subsumption orders relative to a bottom clause. Related work is discussed in Sect. 9. Section 10 concludes the paper.

2 Preliminaries

We assume the reader to be familiar with the basic concepts from logic programming and inductive logic programming (Nienhuys-Cheng and de Wolf 1997) and also the basic concepts from ordered sets and lattices (Davey and Priestley 2002). This section is intended as a brief reminder of some of the concepts used in definitions and theorems in this paper.

Definition 1 (Ordered sets) Let P be a set and R be a binary relation on P. The pair $\langle P, R\rangle$ is said to be:

- a *quasi-ordered set* if R is reflexive and transitive.
- a *partially ordered set* if R is reflexive, transitive and antisymmetric.
- an *equivalence relation* if R is reflexive, transitive and symmetric.

Definition 2 (Mappings between ordered sets) Let $\langle P, \leq\rangle$ and $\langle Q, \subseteq\rangle$ be quasi-ordered sets. A mapping $f : P \to Q$ is said to be:

- *order-preserving* (or *monotone*) if for all x and y in P, $x \leq y$ implies $f(x) \subseteq f(y)$.
- *order-embedding* if for all x and y in P, $x \leq y$ if and only if $f(x) \subseteq f(y)$.
- *order-isomorphism* if it is an order-embedding which maps P onto Q. In this case we say P and Q are (order) isomorphic and write $P \cong Q$.

Remark 1 Let $\langle P, \leq\rangle$ and $\langle Q, \subseteq\rangle$ be partially ordered sets and $f : P \to Q$ be an order-isomorphism. Then we have $x = y$ if and only if $f(x) = f(y)$

Remark 2 Let $\langle P, \leq\rangle$ and $\langle Q, \subseteq\rangle$ be partially ordered sets and $f : P \to Q$ be an order-isomorphism. Then the inverse of f, $f^{-1} : Q \to P$, is also an order-isomorphism.

Definition 3 (lub and glb) Let $\langle P, \leq\rangle$ be a quasi-ordered set and $S \subseteq P$. An element $x \in P$ is an *upper bound* of S if $s \leq x$ for all $s \in S$. An upper bound x of S is a *least upper bound* (*lub*) of S if $x \leq z$ for all upper bounds z of S. Dually, an element $x \in P$ is a *lower bound* of S if $x \leq s$ for all $s \in S$. A lower bound x of S is a *greatest lower bound* (*glb*) of S if $z \leq x$ for all lower bounds z of S.

Definition 4 (Lattice) A quasi-ordered set $\langle L, \leq\rangle$ is called a lattice if for every x and y in L a lub of $\{x, y\}$, also denoted by $x \wedge y$ (read 'x meet y') and a glb of $\{x, y\}$, also denoted by $x \vee y$ (read 'x join y') exist. A lattice $\langle L, \leq\rangle$ is also denoted using the meet and join operators: $\langle L, \wedge, \vee\rangle$.

Definition 5 (Lattice homomorphism) Let $\langle L, \wedge, \vee\rangle$ and $\langle K, \cap, \cup\rangle$ be lattices. A mapping $f : L \to K$ is a *lattice homomorphism* if f is join-preserving and meet-preserving that is for all x and y in L:

1. $f(x \vee y) = f(x) \cup f(y)$ and
2. $f(x \wedge y) = f(x) \cap f(y)$

Definition 6 (Lattice isomorphism) Let $\langle L, \wedge, \vee\rangle$ and $\langle K, \cap, \cup\rangle$ be lattices. A mapping $f : L \to K$ is a *lattice isomorphism* if f is a bijective lattice homomorphism.

Remark 3 If $f : L \to K$ is a one-to-one homomorphism, then the sub-lattice $f(L)$ of K is isomorphic to L and we refer to f as an embedding of L into K.

Remark 4 Let $\langle L, \wedge, \vee\rangle$ and $\langle K, \cap, \cup\rangle$ be lattices.

- a mapping $f : L \to K$ is order-preserving if it is a lattice homomorphism.
- a mapping $f : L \to K$ is order-isomorphism if and only if it is a lattice isomorphism.

Note that if two lattices are isomorphic then for all practical purposes they are identical and differ only in the notation of their elements. In other words, an isomorphism faithfully mirrors the order structure.

The general subsumption order on clauses, also known as θ-subsumption, is defined in the following. First we define substitutions.

Definition 7 (Substitution) A substitution θ is a set $\{v_1/t_1, \ldots, v_n/t_n\}$ where each v_i is a distinct variable and each t_i is a term. We say t_i is substituted for v_i and v_i/t_i is called a binding for v_i. The set $\{v_1, \ldots, v_n\}$ is called the domain of θ, or dom(θ), and $\{t_1, \ldots, t_n\}$ the range of θ, or rng(θ). A substitution $\theta = \{v_1/t_1, \ldots, v_n/t_n\}$ is called a variable substitution if every t_i is a variable. A variable substitution $\theta = \{u_1/v_1, \ldots, u_n/v_n\}$ is said to be a variable renaming if and only if dom(θ) is disjoint from rng(θ) and each v_i is distinct.

Definition 8 (Subsumption on clauses) Let C and D be clauses. We say C subsumes D, denoted by $C \succeq D$, if there exists a substitution θ such that $C\theta$ is a subset of D. C properly subsumes D, denoted by $C \succ D$, if $C \succeq D$ and $D \not\succeq C$. C and D are subsume-equivalent, denoted by $C \sim D$, if $C \succeq D$ and $D \succeq C$.

The subsumption order on atoms, which is a special case of Definition 8, is defined as follows.

Definition 9 (Subsumption on atoms) Let A and B be atoms. We say A subsumes B, denoted by $A \succeq B$, if there exists a substitution θ such that $A\theta = B$. A properly subsumes B, denoted by $A \succ B$, if $A \succeq B$ and $B \not\succeq A$. A and B are subsume-equivalent, denoted by $A \sim B$, if $A \succeq B$ and $B \succeq A$.

Remark 5 Let $\mathcal{A}$ be the set of atoms in a language and $\succeq$ be the subsumption order as defined in Definition 9. Every finite subset of $\mathcal{A}$ has a *most general specialisation* (*mgs*), obtained from the unification algorithm, and a *least general generalisation* (*lgg*), obtained from the anti-unification algorithm. Thus $\langle \mathcal{A}, \succeq \rangle$ is a lattice.

Remark 6 Let $\mathcal{C}$ be a clausal language and $\succeq$ be the subsumption order as defined in Definition 8. Every finite subset of $\mathcal{C}$ has a *most general specialisation* (*mgs*) and a *least general generalisation* (*lgg*). Thus $\langle \mathcal{C}, \succeq \rangle$ is a lattice.

The following definition is a reminder of the concept of refinement operators and several properties of these operators.

Definition 10 (Refinement operator) Let $\mathcal{C}$ be a clausal language and $\succeq$ be the subsumption order as defined in Definition 8. A (downward) *refinement operator* for $\langle \mathcal{C}, \succeq \rangle$ is a function ρ, such that $\rho(C) \subseteq \{D | C \succeq D\}$, for every $C \in \mathcal{C}$.

- The sets of *one-step refinements*, *n-step refinements* and *refinements* of some $C \in \mathcal{C}$ 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 $\rho^*(C) = \rho^1(C) \cup \rho^2(C) \cup \cdots$.
- 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 $C \in \mathcal{C}$, $\rho(C)$ is finite and computable.
- ρ is *proper* if for every $C \in \mathcal{C}$, $\rho(C) \subseteq \{D | C \succ D\}$.

$$\begin{tikzcd}
C: & & p(x,y) \arrow[dl] \arrow[d] \arrow[dr] & & \\
\rho(C): & p(x,x) \arrow[d] & p(x,y) \leftarrow q(x,z) \arrow[dl] \arrow[d] \arrow[dr] & p(x,y) \leftarrow r(w,y) \arrow[dr] & \\
\rho^2(C): & p(x,x) \leftarrow q(x,z) \arrow[d] & p(x,y) \leftarrow q(x,x) \arrow[d] & \begin{array}{c} p(x,y) \leftarrow q(x,z), \\ r(w,y) \end{array} \arrow[d] & \begin{array}{c} p(x,y) \leftarrow r(w,y), \\ q(x,z) \end{array} \arrow[d] \\
\rho^3(C): & p(x,x) \leftarrow q(x,y) \arrow[d] & \begin{array}{c} p(x,y) \leftarrow q(x,x), \\ q(x,y) \end{array} \arrow[d] & \begin{array}{c} p(x,y) \leftarrow q(x,x), \\ r(w,y) \end{array} \arrow[d] & \begin{array}{c} p(x,y) \leftarrow r(y,y), \\ q(x,z) \end{array} \arrow[d] \\
\rho^n(C): & \ldots & \ldots & \ldots & \ldots
\end{tikzcd}$$

$$\rho^*(C) = \rho^0(C) \cup \rho^1(C) \cup \rho^2(C) \ldots$$

Fig. 1 Part of a refinement graph representing (downward) refinements of a clause

- ρ is *complete* if for every $C, D \in \mathcal{C}$ 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* for $\langle \mathcal{C}, \succeq \rangle$ if $\rho^*(\Box) = \mathcal{C}$, where $\Box$ is the top element of $\mathcal{C}$.
- ρ is *non-redundant* if for every $C, D, E \in \mathcal{C}$, $E \in \rho^*(C)$ 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.

Example 1 Figure 1 shows part of a (downward) refinement graph for the subsumption order. In this graph clause $p(x, y)$ is refined either by unifying variables or by adding literals. The refinement operator presented by this graph is not complete as it does not include all possible refinements. It is proper as the graph does not contain cycles. It is redundant because it does not have a tree structure and there is more than one path from $p(x, y)$ to $p(x, x) \leftarrow q(x, z)$.

3 Clause refinement in a Progol-like ILP system

In this section we study clause refinement in ILP systems which consider the hypothesis space bounded by a bottom clause. In particular we discuss refinement in Progol as a representative of these ILP systems. The Progol algorithm (Muggleton 1995) 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 a bottom clause $\bot$ associated with e. In general $\bot$ can have infinite cardinality. Progol uses mode declarations to constrain the search for clauses which subsume $\bot$. Progol's mode declaration (M), definite mode language ($\mathcal{L}(M)$) and depth-bounded mode language ($\mathcal{L}_i(M)$) are defined in Appendix A.

Progol searches a bounded sub-lattice for each example e relative to background knowledge B and mode declarations M. The sub-lattice has a most general element which is the empty clause, $\square$, and a least general element $\bot_i$ which is the most specific element in $\mathcal{L}_i(M)$ such that

$$B \wedge \bot_i \wedge \overline{e} \vdash_h \square$$

where $\vdash_h \square$ denotes derivation of the empty clause in at most h resolutions. The following definition describes a bottom clause $\bot_i$ for a depth-bounded mode language $\mathcal{L}_i(M)$.

Definition 11 (Most-specific clause or bottom clause) Let h, i be natural numbers, B be a set of Horn clauses, $e = a \leftarrow b_1, \ldots, b_n$ be a definite clause, M be a set of mode declarations containing exactly one modeh m such that $a(m) \succeq a$ and $\hat{\bot}$ be the most-specific (potentially infinite) definite clause such that $B \wedge \hat{\bot} \wedge \overline{e} \vdash_h \square$. $\bot_i$ is the most-specific clause in $\mathcal{L}_i(M)$ such that $\bot_i \succeq \hat{\bot}$. C is the most-specific clause in $\mathcal{L}$ if for all C' in $\mathcal{L}$ we have $C' \succeq C$. $\overrightarrow{\bot}$ is $\bot_i$ with a defined ordering over the literals.

In this paper, we refer to $\bot_i$ as $\overrightarrow{\bot}$ or $\bot$ depending on whether we use the ordering of the literals or not. Progol's algorithm for constructing the bottom clause ($\bot_i$) is given in Appendix A.

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 (Muggleton 1995) allows more than one literal in H to be mapped to the same literal l' in $\bot$. However, in Progol's implementation[1] 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 12 (Progol refinement operator ρ_0) Let h, i, B, e, M and $\bot_i$ be defined as in Definition 11 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 $p(v_1, \ldots, v_m)$ 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. $\langle p(v_1, \ldots, v_m), \theta'_m \rangle$ is in $\delta(\theta, k)$ if and only if $l_k = p(u_1, \ldots, u_m)$ 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 splittable then $\theta'_j = \theta'_{j-1} \cup \{v_j/u_j\}$ where v_j is a new variable not in $\mathrm{dom}(\theta'_{j-1})$.

[1] Available from: http://www.doc.ic.ac.uk/~shm/Software/progol4.1/.

$\langle C', \theta', k' \rangle$ 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 $k < n$.

In Definition 12 the variables in $\bot_i$ form a set of equivalences classes over the variables in any clause C which θ-subsumes $\bot_i$. Thus we could write the equivalence class of u in θ as $[v]_u$, the set of all variables in C such that v/u is in θ. The second choice in the definition of δ adds a new variable to an equivalence class $[v_j]_{u_j}$. This will be referred to as *splitting* the variable u_j. Note that in Definition 12 a variable is not splittable if it corresponds to a +type in a modeb since the resulting clause would violate the mode declaration language $\mathcal{L}(M)$ (see Definition 38). In some problems, given enough training examples, the target hypothesis can be learned without variable splitting (using only the variable bindings from the bottom clause). For this reason, the default refinement operators in some Progol-like ILP systems, including Aleph (Srinivasan 2007), do not split variables and only consider adding literals from the bottom clause (i.e. the second choice of δ in Definition 12 is not implemented). However, it can be shown that there are problems where the target hypothesis cannot be found by a Progol-like ILP system without variable splitting. The following is an example where variable splitting is needed.

Example 2 (Variable splitting) Consider learning a half adder logical circuit that performs an addition operation on two binary digits and produces a sum and a carry value which are both binary digits. Suppose M consists of the following mode declarations:

$$\begin{array}{l}\text{modeh}(1, \text{add}(+\text{bin}, +\text{bin}, -\text{bin}, -\text{bin}))\\ \text{modeb}(1, \text{xor}(+\text{bin}, +\text{bin}, -\text{bin}))\\ \text{modeb}(1, \text{and}(+\text{bin}, +\text{bin}, -\text{bin}))\end{array}$$

The type and other background knowledge are defined as follows:

$$B = \left\{\begin{array}{lll}\text{bin}(0) \leftarrow, & \text{bin}(1) \leftarrow, & \text{and}(0,0,0) \leftarrow,\\ \text{and}(0,1,0) \leftarrow, & \text{and}(1,0,0) \leftarrow, & \text{and}(1,1,1) \leftarrow,\\ \text{xor}(0,0,0) \leftarrow, & \text{xor}(0,1,1) \leftarrow, & \text{xor}(1,0,1) \leftarrow,\\ \text{xor}(1,1,0) \leftarrow & & \end{array}\right.$$

The positive and negative examples are as follows:

$$E = \left\{\begin{array}{l}\text{add}(1,0,1,0) \leftarrow, \text{add}(0,0,0,0) \leftarrow, \text{add}(0,1,1,0) \leftarrow,\\ \text{add}(1,1,0,1) \leftarrow, \leftarrow \text{add}(0,1,0,1), \leftarrow \text{add}(1,0,0,1),\\ \leftarrow \text{add}(1,1,1,0), \leftarrow \text{add}(1,1,1,1), \leftarrow \text{add}(0,1,1,1)\end{array}\right.$$

Let $h = 30$ and $i = 3$ and let the first positive example be $e = \text{add}(1,0,1,0) \leftarrow$. In this case $\bot_i$ is as follows:

$$\begin{aligned}\bot_i = \text{add}(A, B, A, B) \leftarrow & \text{xor}(A, A, B), \text{xor}(A, B, A), \text{xor}(B, A, A),\\ & \text{xor}(B, B, B), \text{and}(A, A, A), \text{and}(A, B, B), \text{and}(B, A, B),\\ & \text{and}(B, B, B).\end{aligned}$$

Using the refinement operator in Definition 12, Progol can learn the following target hypothesis:

$$\mathrm{add}(A, B, C, D) \leftarrow \mathrm{xor}(A, B, C), \mathrm{and}(A, B, D).$$

However, this clause cannot be generated from $\bot_i$ without variable splitting, because the bottom clause contains only two distinct variables and yet the target clause contains four variables.

This example represents a group of problems which cannot be learned by a Progol-like ILP system without variable splitting. In general, if the target clause includes a predicate with more than two variables which are defined over a binary domain, then in the bottom clause at least two arguments of this predicate always represent the same variable. This then requires variable splitting in order to generate the target clause from the bottom clause. Progol's refinement operator uses variable splitting by default, however it can be turned off by the user. Aleph's default refinement operator does not implement variable splitting and only considers adding literals from the bottom clause. Variable splitting in Aleph is implemented in an optional setting where equality literals between variables are inserted into the bottom clause to maintain equivalence. However, introducing equality literals in the bottom clause increases the search space considerably and can make the search explore redundant clauses. According to Aleph's manual (Srinivasan 2007), if variable splitting is turned on (*splitvars* is set to true) the bottom clause can be extremely large and probably not practical for large numbers of variable co-references. For the problem mentioned in Example 2, Aleph can find the correct target hypothesis if *splitvars* is set to true. However, in this case Aleph considers a bottom clause with 101 literals compared with the bottom clause with 9 literals considered by Progol.

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 characterise Progol's refinement space. It can be shown that a refinement operator cannot be both complete and non-redundant (Nienhuys-Cheng and de Wolf 1997). 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 (Muggleton 1995). This is demonstrated in the following example.[2]

Example 3 Let B contain definitions for decrementation (dec), addition (plus) and the clause $\mathrm{mult}(0, X, 0) \leftarrow$ with appropriate mode declarations M and let the example e be the clause $\mathrm{mult}(1, 1, 1) \leftarrow$. Then $\bot$ is the clause

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

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 $\bot$ there will be no element of Progol's $\rho^*(\Box)$ containing this clause or a subsume-equivalent of this clause.

[2] This example is a corrected version of Example 30 in (Muggleton 1995).

Badea and Stanciu (1999) describe two types of Progol's incompleteness. This example is related to the first type of incompleteness which 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 $\bot$ can only be considered from left to right. Moreover, each literal from $\bot$ 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 4 Let C, D and $\bot$ be clauses as defined below

$$C = \mathrm{p}(X, Y) \leftarrow \mathrm{q}(X, X), \mathrm{q}(Y, W),$$

$$D = \mathrm{p}(X, Y) \leftarrow \mathrm{q}(Z, X), \mathrm{q}(Y, Y),$$

$$\bot = \mathrm{p}(X, Y) \leftarrow \mathrm{q}(X, X), \mathrm{q}(Y, Y).$$

C and D can be generated by Progol's refinement (i.e. $C, D \in \rho^*(\square)$), however, clause E below which is the least general generalisation (*lgg*) of C and D cannot be generated (i.e. $E \notin \rho^*(\square)$).

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

Example 4 is related to the second type of incompleteness which is due to the fact that each literal from $\bot$ can be selected only once. Clause E, therefore, cannot be in $\rho^*(\square)$ as this will require more than one literal of E to be mapped to the same literal of $\bot$. As another example of the second type of incompleteness, consider the following example adopted from Badea and Stanciu (1999).

Example 5 Let $\bot = \mathrm{p}(X) \leftarrow \mathrm{q}(X, X)$, then Progol's refinement only considers the following hypotheses.

$$C = \mathrm{p}(X),$$

$$D = \mathrm{p}(X) \leftarrow \mathrm{q}(X, Y),$$

$$E = \mathrm{p}(X) \leftarrow \mathrm{q}(X, X).$$

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

$$C'_1 = \mathrm{p}(X) \leftarrow \mathrm{q}(X, Y), \mathrm{q}(Y, X),$$

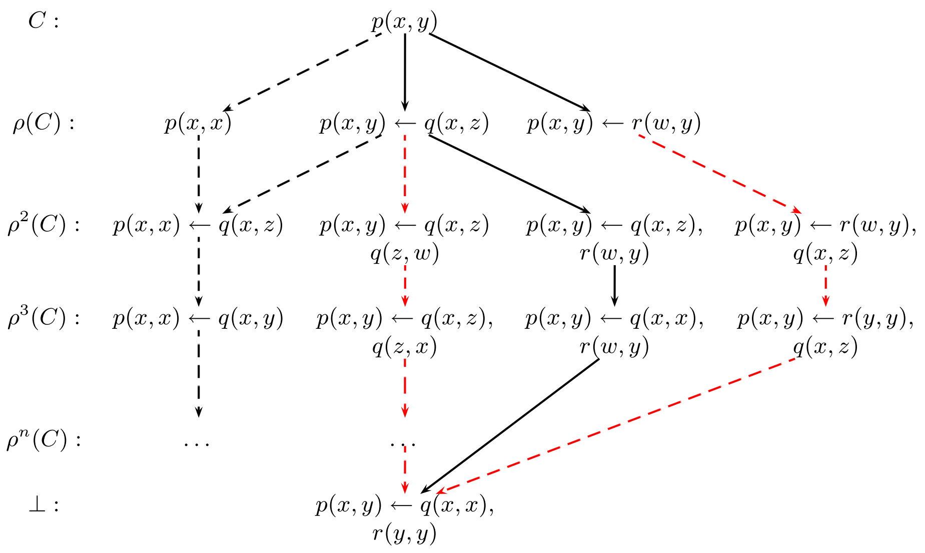
$$C'_2 = \mathrm{p}(X) \leftarrow \mathrm{q}(X, Y), \mathrm{q}(Y, Z), \mathrm{q}(Z, X),$$

$$C'_3 = \mathrm{p}(X) \leftarrow \mathrm{q}(X, Y), \mathrm{q}(Y, Z), \mathrm{q}(Z, W), \mathrm{q}(W, X),$$

$$\ldots.$$

In this example clause C'_n can be constructed only if more than one literal (i.e. $n + 1$ literals) from C'_n could be mapped to the same literal $\mathrm{q}(X, X)$ from $\bot$ (which is not allowed in Progol's refinement).

Figure 2 summarises the two types of incompleteness discussed in this section. This figure shows part of a refinement graph bounded by a bottom clause as in Progol. Suppose



$C:$
$p(x,y)$
$\rho(C):$
$p(x,x)$
$p(x,y) \leftarrow q(x,z)$
$p(x,y) \leftarrow r(w,y)$
$\rho^2(C):$
$p(x,x) \leftarrow q(x,z)$
$p(x,y) \leftarrow q(x,z)$ $q(z,w)$
$p(x,y) \leftarrow q(x,z),$ $r(w,y)$
$p(x,y) \leftarrow r(w,y),$ $q(x,z)$
$\rho^3(C):$
$p(x,x) \leftarrow q(x,y)$
$p(x,y) \leftarrow q(x,z),$ $q(z,x)$
$p(x,y) \leftarrow q(x,x),$ $r(w,y)$
$p(x,y) \leftarrow r(y,y),$ $q(x,z)$
$\rho^n(C):$
$\perp:$
$p(x,y) \leftarrow q(x,x),$ $r(y,y)$

Fig. 2 Part of a refinement graph bounded by a bottom clause as in Progol. *Dashed lines* represent refinement steps which are not considered by Progol's refinement. *Red dashed lines* represent missing refinement steps which could lead to incompleteness with respect to general subsumption

that the bottom clause $\perp$ is given by $p(x, y) \leftarrow q(x, x), r(y, y)$. Dashed lines represent refinement steps which are not considered by Progol's refinement. For example, refinement step from $p(x, y)$ to $p(x, x)$ is not considered because according to Progol's refinement operator, a literal is not allowed to be more specific than the corresponding literal from the bottom clause (i.e. $p(x, y)$ in this case). Red dashed lines represent missing refinement steps which could lead to incompleteness with respect to general subsumption. For example, refinement step from $p(x, y) \leftarrow r(w, y)$ to $p(x, y) \leftarrow r(w, y), q(x, z)$ is not considered by Progol's refinement due to the choice of ordering in the bottom clause. In this example a subsume-equivalent of clause $p(x, y) \leftarrow r(w, y), q(x, z)$ (i.e. $p(x, y) \leftarrow q(x, z), r(w, y)$) appears in the refinement graph. However, as shown in example 3, the choice of ordering in the bottom clause and the variable dependencies in the literals could lead to incompleteness (first type of incompleteness). Moreover, refinement step from $p(x, y) \leftarrow q(x, z)$ to $p(x, y) \leftarrow q(x, z), q(z, x)$ is missing because each literal from the bottom clause can be selected only once (second type of incompleteness).

It has been suggested (Badea and Stanciu 1999) 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, Badea and Stanciu (1999) 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 $\perp = \mathrm{p}(X) \leftarrow \mathrm{q}(X, X)$ 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 $\perp$ in Example 3. C weakly subsumes $\perp$ but clause C is not considered by Progol's refinement.

As mentioned before, Progol's refinement operator scans $\perp$ from left to right and for each literal l' of $\perp$ 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 show how Progol's refinement can be characterised using sequential subsumption.

4 Ordered clauses and sequential subsumption

In this section we define the concepts of ordered clauses and sequential subsumption which will be used for characterising clause refinement in a Progol-like ILP system. According to Definition 38, clauses which are considered by Progol's refinement (i.e. clauses in $\mathcal{L}(M)$) are defined with a total ordering over the literals. In order to characterise Progol's refinement we adopt an explicit representation for ordered clauses. The concept of ordered clauses has been used before 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 in the standard representation of clauses (which use a set notation) and therefore ordered clauses are used instead (Nienhuys-Cheng and de Wolf 1997). A subsumption relation for ordered clauses is studied in Kuwabara et al. (2006). The difference between this subsumption order and the subsumption order considered in this paper is discussed in Sect. 9. There are also other applications of ordered clauses and sequential subsumption, for example in the context of data mining from sequential data (e.g. Lee and De Raedt 2003). In this paper we use the same notion used in Nienhuys-Cheng and de Wolf (1997) 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 13 (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.

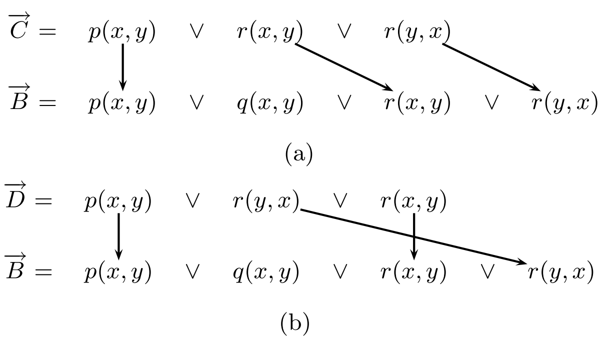
Unlike conventional clauses, the order and duplication of literals matter for ordered clauses. For example, $\overrightarrow{C} = p(X) \vee \neg q(X)$, $\overrightarrow{D} = \neg q(X) \vee p(X)$ and $\overrightarrow{E} = p(X) \vee \neg q(X) \vee p(X)$ are different ordered clauses while they all correspond 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 (Plotkin 1971). However, in this paper we use selections to define mappings of literals between two ordered clauses.

Definition 14 (Compatible literals) Literals L and M are compatible if they have the same sign and predicate symbol.

Definition 15 (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 16 (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 $\{1, 2, \ldots, m\}$.



$\overrightarrow{C} = p(x,y) \vee r(x,y) \vee r(y,x)$
$\overrightarrow{B} = p(x,y) \vee q(x,y) \vee r(x,y) \vee r(y,x)$
(a)
$\overrightarrow{D} = p(x,y) \vee r(y,x) \vee r(x,y)$
$\overrightarrow{B} = p(x,y) \vee q(x,y) \vee r(x,y) \vee r(y,x)$
(b)

Fig. 3 (**a**) $\overrightarrow{C}$ is a subsequence of $\overrightarrow{B}$ because there exists strictly increasing selection function $s_1 = \{(1,1),(2,3),(3,4)\}$ which maps each literal from $\overrightarrow{C}$ to an equivalent literal from $\overrightarrow{B}$ (**b**) $\overrightarrow{D}$ is not a subsequence of $\overrightarrow{B}$

Example 6 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 17 (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 $\overrightarrow{C} \sqsubseteq \overrightarrow{D}$, if there exists a strictly increasing selection function $s \subseteq \{1,\dots,l\} \times \{1,\dots,m\}$ such that for each $(i,j) \in s$, $L_i = M_j$.

Example 7 In Fig. 3, $\overrightarrow{C}$ is a subsequence of $\overrightarrow{B}$ because there exists increasing selection function $s_1 = \{(1,1),(2,3),(3,4)\}$ which maps literals from $\overrightarrow{C}$ to equivalent literals from $\overrightarrow{D}$. However, $\overrightarrow{D}$ is not a subsequence of $\overrightarrow{B}$ because an increasing selection function does not exist for $\overrightarrow{D}$ and $\overrightarrow{B}$.

Definition 18 (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 19 (Sequential subsumption) Let $\overrightarrow{C}$ and $\overrightarrow{D}$ be ordered clauses. We say $\overrightarrow{C}$ sequentially subsumes $\overrightarrow{D}$, 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 8 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 $C \succeq D$.*

Table 1 A comparison between ordered clauses and conventional clauses

Ordered clauses	Conventional clauses
$\overrightarrow{C} = L_1 \vee L_2 \vee \cdots \vee L_n$	$C = \{L_1, L_2, \ldots, L_n\}$
Mapping of literals (s)	Undefined
Subsequence ($\sqsubseteq$)	Subset ($\subseteq$)
Sequential subsumption ($\succeq_s$)	Subsumption ($\succeq$)

Proof Suppose $\overrightarrow{C} \succeq_s \overrightarrow{D}$, then according to Definition 19 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 8, the converse of Theorem 1 does not hold in general. Table 1 shows a comparison between corresponding concepts for ordered clauses and conventional clauses.

5 Subsumption order relative to a bottom clause

As shown in Sect. 3, clause refinement in Progol-like ILP systems cannot be described by the general subsumption order. In this section we define a subsumption order relative to $\bot$ (i.e. $\succeq_\bot$) which can capture clause refinement in these systems. First we define $\overrightarrow{\mathcal{L}}_\bot$ as the set of definite ordered clauses which are sequential generalisation of $\overrightarrow{\bot}$. In this section we show that the refinement space of a Progol-like ILP system can be characterised using $\overrightarrow{\mathcal{L}}_\bot$.

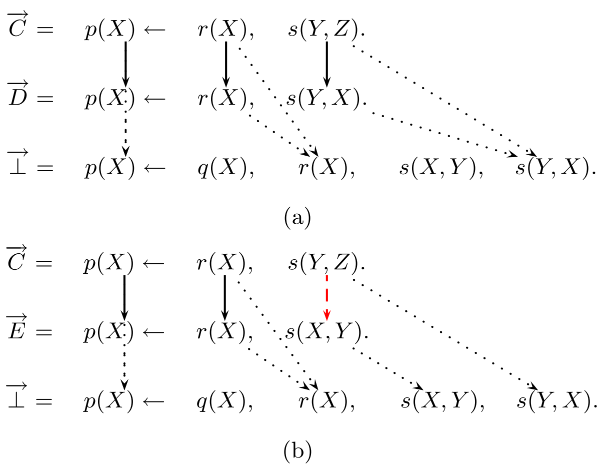
Definition 20 ($\overrightarrow{\mathcal{L}}_\bot$) Let $\overrightarrow{\bot}$ be the bottom clause as defined in Definition 11 and $\overrightarrow{C}$ a definite ordered clause. $\overrightarrow{C}$ is in $\overrightarrow{\mathcal{L}}_\bot$ if and only if there exists a substitution θ such that $\overrightarrow{C}\theta$ is a subsequence of $\overrightarrow{\bot}$.

Let us consider the examples in Sect. 3 with respect to Definition 20. In Example 3, if we now consider $\overrightarrow{C}$ and $\overrightarrow{\bot}$ as ordered clauses then $\overrightarrow{C} \notin \overrightarrow{\mathcal{L}}_\bot$, because there is no substitution θ such that $\overrightarrow{C}\theta$ can be a subsequence of $\overrightarrow{\bot}$. Similarly in Example 5, $\overrightarrow{C'_1} \notin \overrightarrow{\mathcal{L}}_\bot$, etc.

As in Muggleton (1995), the languages which we consider in this paper (e.g. $\overrightarrow{\mathcal{L}}_\bot$) correspond to a set of clauses which have the characteristics of generalisations of a flattened bottom clause. Therefore, all clauses in $\overrightarrow{\mathcal{L}}_\bot$ can be treated as if they were function-free and all substitutions we consider are variable substitutions. The following example demonstrates how clauses with function symbols can be dealt with as though they were function-free by using *flattening* (Rouveirol 1992).

Example 9 The clause $D = \text{nat}(s(s(X))) \leftarrow \text{nat}(X)$ can be flattened to the function-free clause $D' = \text{nat}(V) \leftarrow s(V, W), s(W, X), \text{nat}(X)$ where s is defined as $s(X, s(X))$.

In Definition 20, the sequential subsumption is used to define the hypotheses language $\overrightarrow{\mathcal{L}}_\bot$. In this section we prove that Progol's refinement space can be characterised by $\overrightarrow{\mathcal{L}}_\bot$. However, Progol's refinement operator cannot be characterised by sequential subsumption and we need to define a subsumption order relative to a bottom clause. This is because only literals are comparable with respect to Progol's refinement which correspond to the same



$\overrightarrow{C} = p(X) \leftarrow r(X), s(Y,Z).$
$\overrightarrow{D} = p(X) \leftarrow r(X), s(Y,X).$
$\overrightarrow{\bot} = p(X) \leftarrow q(X), r(X), s(X,Y), s(Y,X).$
(a)
$\overrightarrow{C} = p(X) \leftarrow r(X), s(Y,Z).$
$\overrightarrow{E} = p(X) \leftarrow r(X), s(X,Y).$
$\overrightarrow{\bot} = p(X) \leftarrow q(X), r(X), s(X,Y), s(Y,X).$
(b)

Fig. 4 Comparable literals with respect to a bottom clause (**a**) literals from $\overrightarrow{C}$ are comparable with the literals from $\overrightarrow{D}$ respectively as they are mapped to the same literals from $\overrightarrow{\bot}$ (**b**) the third literal from $\overrightarrow{C}$ and the third literal from $\overrightarrow{E}$ are not comparable as they are mapped to different literals from $\overrightarrow{\bot}$

literals of $\overrightarrow{\bot}$. According to the definition of Progol's refinement operator (Definition 12), refinements of a clause are constructed by adding literals which are generalisations of a literal from $\overrightarrow{\bot}$. These literals are generated by δ and they all correspond to the same literal l_k from $\overrightarrow{\bot}$. 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 $\overrightarrow{\bot}$. This has been demonstrated in the following example.

Example 10 Let $\overrightarrow{\bot} = p(X) \leftarrow q(X), r(X), s(X,Y), s(Y,X)$ be an ordered clause and $\overrightarrow{C} = p(X) \leftarrow r(X), s(Y,Z)$ and $\overrightarrow{D} = p(X) \leftarrow r(X), s(Y,X)$ be ordered clauses in $\overrightarrow{\mathcal{L}}_\bot$ as shown in Fig. 4. Suppose that the literals of $\overrightarrow{C}$ are mapped to the first, the third and the fifth literals of $\overrightarrow{\bot}$ respectively, and similarly the literals of $\overrightarrow{D}$ are mapped to the first, the second, the third and the fifth literals of $\overrightarrow{\bot}$ as in Fig. 4.a. In this case, literals from $\overrightarrow{C}$ are comparable with the first, the third and the fourth literals from $\overrightarrow{D}$ respectively. However, in Fig. 4.b, the third literal from $\overrightarrow{C}$ and the third literal from $\overrightarrow{E}$ are mapped to different literals from $\overrightarrow{\bot}$ and therefore they are not comparable with respect to Progol's refinement (though they are comparable with respect to general subsumption or sequential subsumption).

In the following we define a subsumption order relative to $\bot$ (i.e. $\succeq_\bot$) which can capture Progol's refinement. First we define subsequence relative to $\bot$.

Definition 21 (Subsequence relative to $\bot$) Let $\overrightarrow{\bot}$ and $\overrightarrow{\mathcal{L}}_\bot$ be as defined in Definition 20 and $\overrightarrow{C}$ and $\overrightarrow{D}$ be ordered clauses in $\overrightarrow{\mathcal{L}}_\bot$ such that the selection function s_1 maps literals from $\overrightarrow{C}\theta_1$ to equivalent literals from $\overrightarrow{\bot}$ and the selection function s_2 maps literals from $\overrightarrow{D}\theta_2$ to equivalent literals from $\overrightarrow{\bot}$. $\overrightarrow{C}$ is a subsequence of $\overrightarrow{D}$ relative to $\bot$, denoted by $\overrightarrow{C} \sqsubseteq_\bot \overrightarrow{D}$, if $\overrightarrow{C}$ is a subsequence of $\overrightarrow{D}$ and $\text{rng}(s_1) \subseteq \text{rng}(s_2)$, where $\text{rng}(f)$ is the range of function f.

In Definition 21 each literal L_i from $\overrightarrow{C}$ is mapped to an equivalent literal M_j from $\overrightarrow{D}$ and L_i and M_j both correspond to the same literal from $\overrightarrow{\bot}$.

Definition 22 (Subsumption relative to $\bot$) Let $\overrightarrow{\bot}$ and $\overrightarrow{\mathcal{L}}_\bot$ be as defined in Definition 20 and $\overrightarrow{C}$ and $\overrightarrow{D}$ be ordered clauses in $\overrightarrow{\mathcal{L}}_\bot$. We say $\overrightarrow{C}$ subsumes $\overrightarrow{D}$ relative to $\bot$, 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 $\perp$. $\overrightarrow{C}$ is a proper generalisation of $\overrightarrow{D}$ relative to $\perp$, denoted by $\overrightarrow{C} \succ_\perp \overrightarrow{D}$, if $\overrightarrow{C} \succeq_\perp \overrightarrow{D}$ and $\overrightarrow{D} \not\succeq_\perp \overrightarrow{C}$. $\overrightarrow{C}$ and $\overrightarrow{D}$ are equivalent with respect to subsumption relative to $\perp$, denoted by $\overrightarrow{C} \sim_\perp \overrightarrow{D}$, if $\overrightarrow{C} \succeq_\perp \overrightarrow{D}$ and $\overrightarrow{D} \succeq_\perp \overrightarrow{C}$.

Example 11 Let $\overrightarrow{\perp}$, $\overrightarrow{C}$, $\overrightarrow{D}$ and $\overrightarrow{E}$ be ordered clauses in Example 10 as shown in Fig. 4. Then $\overrightarrow{C}$ subsumes $\overrightarrow{D}$ relative to $\perp$ since there is a substitution θ such that $\overrightarrow{C}\theta$ is a subsequence of $\overrightarrow{D}$ relative to $\perp$. However, $\overrightarrow{C}$ does not subsume $\overrightarrow{E}$ relative to $\perp$ since there is no substitution θ such that $\overrightarrow{C}\theta$ can be a subsequence of $\overrightarrow{E}$ relative to $\perp$. This is because the third literal from $\overrightarrow{C}$ and the third literal from $\overrightarrow{E}$ are mapped to different literals from $\overrightarrow{\perp}$.

In the following we first define $\overrightarrow{\mathcal{L}}_\perp(M)$ by analogy to Progol's $\mathcal{L}_i(M)$ and then prove that Progol's refinement operator, ρ_0, is weakly complete for $\langle \overrightarrow{\mathcal{L}}_\perp(M), \succeq_\perp\rangle$. As shown by the examples from Sect. 3, in particular Example 3, Progol's refinement cannot be complete or even weakly complete for general subsumption order. However, it can be weakly complete for $\langle \overrightarrow{\mathcal{L}}_\perp(M), \succeq_\perp\rangle$.

Definition 23 ($\overrightarrow{\mathcal{L}}_\perp(M)$) Let $\overrightarrow{\mathcal{L}}_\perp$ and $\mathcal{L}_i(M)$ be as defined in Definitions 20 and 40 respectively. $\overrightarrow{C}$ is in $\overrightarrow{\mathcal{L}}_\perp(M)$ if and only if $\overrightarrow{C}$ is in $\overrightarrow{\mathcal{L}}_\perp$ and C is in $\mathcal{L}_i(M)$.

In Definition 12, the refinement operator ρ_0, as ρ in Muggleton (1995), is defined for clauses in $\mathcal{L}_i(M)$. However, ρ_0 can be also defined for clauses in $\overrightarrow{\mathcal{L}}_\perp(M)$ if we let C and C' to be ordered clauses in $\overrightarrow{\mathcal{L}}_\perp(M)$ and $\overrightarrow{C}\theta$ be a subsequence (rather than a subset) of the bottom clause. In this case, it can be shown that ρ_0 is weakly complete for $\langle \overrightarrow{\mathcal{L}}_\perp(M), \succeq_\perp\rangle$. This theorem also suggests that refinement space in a Progol-like ILP system can be characterised by $\overrightarrow{\mathcal{L}}_\perp(M)$.

Lemma 1 *Let $\delta(\theta, k)$ be as defined in Definition 12 and $\overrightarrow{C}$ and $\overrightarrow{C'} = \overrightarrow{C} \vee l$ be ordered clauses in $\overrightarrow{\mathcal{L}}_\perp(M)$ such that $\overrightarrow{C}\theta$ and $\overrightarrow{C'}\theta'$ be subsequences of $\overrightarrow{\perp}$ and $l\theta' = l_k$ where l_k is the kth literal of $\overrightarrow{\perp}$. Then, there exists $\langle l', \theta''\rangle$ in $\delta(\theta, k)$ such that l and l' are variants.*

Proof Let literals l_k, l and l' be denoted simply by $p(u_1, \ldots, u_m)$, $p(v_1, \ldots, v_m)$ and $p(v'_1, \ldots, v'_m)$ respectively, despite the sign and function symbols (as in Definition 12). Let $l_k = p(u_1, \ldots, u_m)$, $l = p(v_1, \ldots, v_m)$ and $l' = p(v'_1, \ldots, v'_m)$. We have $p(v_1, \ldots, v_m)\theta' = p(u_1, \ldots, u_m)$. We show that $\langle l', \theta''\rangle$ can be constructed using $\delta(\theta, k)$ and there exist variables $v'_1, \ldots, v'_m$ and substitution θ'' such that $p(v'_1, \ldots, v'_m)\theta'' = p(u_1, \ldots, u_m)$ and l and l' are variants. Let $\theta''_0 = \theta$ and for each $v_j/u_j \in \theta'$ where $1 \le j \le m$ if v_j is a new variable with respect to $\{v_1, \ldots, v_{j-1}\}$ and u_j is splittable then, using choice 2 in the definition of δ, $\theta''_j = \theta''_{j-1} \cup \{v'_j/u_j\}$ where v'_j is a new variable not in $dom(\theta''_{j-1})$. Otherwise, by using choice 1 in the definition of δ, $\theta''_j = \theta''_{j-1}$. By construction, $\langle l', \theta''_m\rangle$ is in $\delta(\theta, k)$ and there is a one-to-one mapping between variables v'_j and v_j for $1 \le j \le m$. Variable substitutions $\sigma_1 = \{v_1/v'_1, \ldots, v_m/v'_m\}$ and $\sigma_2 = \{v'_1/v_1, \ldots, v'_m/v_m\}$ are therefore variable renamings. Hence, $l\sigma_1 = l'$ and $l'\sigma_2 = l$ and therefore l and l' are variants. □

Theorem 2 *In Definition 12 let $\overrightarrow{C}$ and $\overrightarrow{C'}$ be ordered clauses in $\overrightarrow{\mathcal{L}}_\perp(M)$ and $\overrightarrow{C}\theta$ be a subsequence (rather than a subset) of the bottom clause. Then ρ_0 is weakly complete for $\langle \overrightarrow{\mathcal{L}}_\perp(M), \succeq_\perp\rangle$.*

Proof We need to show that $\rho_0^*(\langle\Box,\emptyset,1\rangle) = \overrightarrow{\mathcal{L}}_\perp(M)$. We show that for each $\overrightarrow{C} \in \overrightarrow{\mathcal{L}}_\perp(M)$, 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, $0 \le j < i$, that the lemma is true, we will show that it is also true for $j+1$. Suppose the lemma is true for j, this implies that there is a ρ_0-chain from $\Box$ to an alphabetical variant of $\overrightarrow{C}_j$ such that $\overrightarrow{C}_j$ is an ordered clause in $\overrightarrow{\mathcal{L}}_\perp(M)$ with j literals added from $\overrightarrow{C}$. Therefore, there is a substitution θ such that $\overrightarrow{C}_j\theta$ is a subsequence of $\overrightarrow{\perp}$ and we assume that the j-th literal of $\overrightarrow{C}_j$ is mapped to the k-th literal of $\overrightarrow{\perp}$. 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 $\overrightarrow{\perp}$, where $k < k'$ (because $\overrightarrow{C}_j$ and $\overrightarrow{C}_{j+1}$ are sequential generalisations of $\overrightarrow{\perp}$). 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 choice 2 in the definition of ρ_0 in order to skip $k'-k$ literals of $\overrightarrow{\perp}$. According to Lemma 1, there exists $\langle l',\theta'\rangle$ in $\delta(\theta,k')$ such that l and l' are variants. Therefore, by selecting choice 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 variant of $\overrightarrow{C}_{j+1}$ and this completes the proof. $\Box$

6 The lattice structure of the subsumption order relative to $\perp$

In the following we study the ordered set defined by $\overrightarrow{\mathcal{L}}_\perp$ and the subsumption order relative to $\perp$ and show that $\langle\overrightarrow{\mathcal{L}}_\perp, \succeq_\perp\rangle$ is a lattice. First we show that $\succeq_\perp$ is a quasi-order and then we prove that each pair of ordered clauses in $\overrightarrow{\mathcal{L}}_\perp$ have a most general specialisation (mgs$_\perp$) and a least general generalisation (lgg$_\perp$) in $\overrightarrow{\mathcal{L}}_\perp$.

Lemma 2 *Subsequence relation relative to $\perp$ ($\sqsubseteq_\perp$) is transitive.*

Proof Let $\overrightarrow{C}$, $\overrightarrow{D}$ and $\overrightarrow{E}$ be ordered clauses in $\overrightarrow{\mathcal{L}}_\perp$ such that $\overrightarrow{C}\theta_1 \sqsubseteq \overrightarrow{\perp}$, $\overrightarrow{D}\theta_2 \sqsubseteq \overrightarrow{\perp}$ and $\overrightarrow{E}\theta_3 \sqsubseteq \overrightarrow{\perp}$ where selection functions s_1, s_2 and s_3 map literals from $\overrightarrow{C}\theta_1$, $\overrightarrow{D}\theta_2$ and $\overrightarrow{E}\theta_3$ to equivalent literals from $\overrightarrow{\perp}$ respectively. Let $\overrightarrow{C} \sqsubseteq_\perp \overrightarrow{D}$ and $\overrightarrow{D} \sqsubseteq_\perp \overrightarrow{E}$. Then according to Definition 21 there exist strictly increasing selection functions s_4 and s_5 such that s_4 maps literals from $\overrightarrow{C}$ to equivalent literals from $\overrightarrow{D}$ and s_5 maps literals from $\overrightarrow{D}$ to equivalent literals from $\overrightarrow{E}$ and $\mathrm{rng}(s_1) \subseteq \mathrm{rng}(s_2) \subseteq \mathrm{rng}(s_3)$. Then $s = s_4 \circ s_5$ is also a strictly increasing function which maps literals from $\overrightarrow{C}$ to equivalent literals from $\overrightarrow{E}$ and $\mathrm{rng}(s_1) \subseteq \mathrm{rng}(s_3)$. This implies $\overrightarrow{C} \sqsubseteq_\perp \overrightarrow{E}$. $\Box$

Theorem 3 *Subsumption order relative to $\perp$ ($\succeq_\perp$) is a quasi-order.*

Proof For every ordered clause $\overrightarrow{C}$, we have $\overrightarrow{C} \succeq_\perp \overrightarrow{C}$. The relation $\succeq_\perp$ is therefore reflexive. Let $\overrightarrow{C}$, $\overrightarrow{D}$ and $\overrightarrow{E}$ be ordered clauses such that $\overrightarrow{C} \succeq_\perp \overrightarrow{D}$ and $\overrightarrow{D} \succeq_\perp \overrightarrow{E}$. Thus, there exist substitutions θ_1 and θ_2 such that $\overrightarrow{C}\theta_1 \sqsubseteq_\perp \overrightarrow{D}$ and $\overrightarrow{D}\theta_2 \sqsubseteq_\perp \overrightarrow{E}$. We have $\overrightarrow{C}\theta_1\theta_2 \sqsubseteq \overrightarrow{D}\theta_2 \sqsubseteq \overrightarrow{E}$ and then according to Lemma 2 $\overrightarrow{C}\theta_1\theta_2$ is a subsequence of $\overrightarrow{E}$ relative to $\perp$ and therefore $\overrightarrow{C} \succeq_\perp \overrightarrow{E}$. The relation $\succeq_\perp$ is reflexive and transitive and therefore it is a quasi-order. $\Box$

In the following we prove that each pair of ordered clauses in $\overrightarrow{\mathcal{L}}_\perp$ have *mgs* and *lgg*. As in Nienhuys-Cheng and de Wolf (1997) we use a sequence of pairs of compatible literals (i.e.

selections) to bridge between the definitions of *mgs* and *lgg* for atoms and the definitions of *mgs* and *lgg* for clauses. The following definition is similar to the one used in Nienhuys-Cheng and de Wolf (1997) adopted for subsumption relative to a bottom clause.

Definition 24 Let $\overrightarrow{C}$ and $\overrightarrow{D}$ be ordered clauses and $S = (L_1, M_1), \ldots, (L_n, M_n)$ a sequence of (not necessarily all) pairs of compatible literals from $\overrightarrow{C}$ and $\overrightarrow{D}$ relative to $\bot$ such that $L_1 \vee L_2 \vee \cdots \vee L_n$ is a subsequence of $\overrightarrow{C}$ relative to $\bot$ and $M_1 \vee M_2 \vee \cdots \vee M_n$ is a subsequence of $\overrightarrow{D}$ relative to $\bot$. Then we let $\overrightarrow{C_S} = L_1 \vee L_2 \vee \cdots \vee L_n$ and $\overrightarrow{D_S} = M_1 \vee M_2 \vee \cdots \vee M_n$.

Lemma 3 *Let $\overrightarrow{C}$ and $\overrightarrow{D}$ be ordered clauses in $\overrightarrow{\mathcal{L}}_\bot$ and S, $\overrightarrow{C_S}$ and $\overrightarrow{D_S}$ be as defined in Definition 24. Then, $\overrightarrow{C_S}$ and $\overrightarrow{D_S}$ are in $\overrightarrow{\mathcal{L}}_\bot$.*

Proof $\overrightarrow{C}$ and $\overrightarrow{D}$ are ordered clauses in $\overrightarrow{\mathcal{L}}_\bot$ and there exist variable substitutions θ_1 and θ_2 such that $\overrightarrow{C}\theta_1$ and $\overrightarrow{D}\theta_2$ are subsequences of $\overrightarrow{\bot}$. But according to Definition 24, $\overrightarrow{C_S}$ is a subsequence of $\overrightarrow{C}$ and $\overrightarrow{D_S}$ is a subsequence of $\overrightarrow{D}$. Then according to Lemma 2, $\overrightarrow{C_S}\theta_1$ and $\overrightarrow{D_S}\theta_2$ are subsequences of $\overrightarrow{\bot}$. Thus, $\overrightarrow{C_S}$ and $\overrightarrow{D_S}$ are in $\overrightarrow{\mathcal{L}}_\bot$. □

Lemma 4 *Let $\overrightarrow{C}$ and $\overrightarrow{D}$ be ordered clauses in $\overrightarrow{\mathcal{L}}_\bot$ and S, $\overrightarrow{C_S}$ and $\overrightarrow{D_S}$ be as defined in Definition 24. Then, $\overrightarrow{C_S}$ and $\overrightarrow{D_S}$ are unifiable.*

Proof According to Definition 24, for each literal L_i from $\overrightarrow{C_S} = L_1 \vee L_2 \vee \cdots \vee L_n$ there exists a corresponding compatible literal M_i from $\overrightarrow{D_S} = M_1 \vee M_2 \vee \cdots \vee M_n$. According to Lemma 3, there exist variable substitutions θ_1 and θ_2 such that $\overrightarrow{C_S}\theta_1$ and $\overrightarrow{D_S}\theta_2$ are subsequences of $\overrightarrow{\bot}$. Let $\overrightarrow{E} = N_1 \vee N_2 \vee \cdots \vee N_n$ be a subsequence of $\overrightarrow{\bot}$ such that $\overrightarrow{C_S}\theta_1 = \overrightarrow{E}$ and $\overrightarrow{D_S}\theta_2 = \overrightarrow{E}$. Let $\theta = \theta_1 \cup \theta_2$ then we have $\overrightarrow{C_S}\theta = \overrightarrow{D_S}\theta = \overrightarrow{E}$. Thus, $\overrightarrow{C_S}$ and $\overrightarrow{D_S}$ are unifiable and θ is a unifier for $\{\overrightarrow{C_S}, \overrightarrow{D_S}\}$. □

Lemma 5 *Let $\overrightarrow{C}$ be an ordered clauses in $\overrightarrow{\mathcal{L}}_\bot$ and $\overrightarrow{D}$ is derived from $\overrightarrow{C}$ by removing some literals without changing the order of the remaining literals. Then, $\overrightarrow{D}$ is a subsequence of $\overrightarrow{C}$ relative to $\bot$.*

Proof $\overrightarrow{C}$ is in $\overrightarrow{\mathcal{L}}_\bot$ and therefore there are substitution θ_1 and a strictly increasing selection function s_1 which maps literals of $\overrightarrow{C}\theta_1$ to equivalent literals from $\overrightarrow{\bot}$. $\overrightarrow{D}$ is derived from $\overrightarrow{C}$ by removing some literals. Thus, $\overrightarrow{D}$ is a subsequence of $\overrightarrow{C}$ and there are substitution $\theta_2 \subseteq \theta_1$ and a selection function $s_2 \subseteq s_1$ which maps literals of $\overrightarrow{D}\theta_2$ to equivalent literals from $\overrightarrow{\bot}$. $s_2 \subseteq s_1$ implies that θ_2 is a strictly increasing selection function and that $\mathrm{rng}(s_2) \subseteq \mathrm{rng}(s_1)$. Then $\overrightarrow{D}$ is in $\overrightarrow{\mathcal{L}}_\bot$ and according to Definition 21, $\overrightarrow{D}$ is a subsequence of $\overrightarrow{C}$ relative to $\bot$. □

Theorem 4 (Existence of $\mathrm{mgs}_\bot$ in $\overrightarrow{\mathcal{L}}_\bot$) *For every ordered clauses $\overrightarrow{C}$ and $\overrightarrow{D}$ in $\overrightarrow{\mathcal{L}}_\bot$, there exists an* $\mathrm{mgs}_\bot$ *of $\overrightarrow{C}$ and $\overrightarrow{D}$ in $\overrightarrow{\mathcal{L}}_\bot$.*

Proof $\overrightarrow{C}$ and $\overrightarrow{D}$ are ordered clauses in $\overrightarrow{\mathcal{L}}_\bot$ and there exist variable substitutions θ_1 and θ_2 such that $\overrightarrow{C}\theta_1$ and $\overrightarrow{D}\theta_2$ are subsequences of $\overrightarrow{\bot}$. Let S, $\overrightarrow{C_S}$ and $\overrightarrow{D_S}$ be as defined in Definition 24 such that S is a sequence of all pairs of compatible literals from $\overrightarrow{C}$ and $\overrightarrow{D}$ relative to $\bot$. According to Lemma 4, $\overrightarrow{C_S}$ and $\overrightarrow{D_S}$ are unifiable. Let σ be an *mgu* for $\{\overrightarrow{C_S}, \overrightarrow{D_S}\}$, $\theta = \{\theta_1 \cup \theta_2\}$, n be the number of literals in $\overrightarrow{\bot}$ and $\overrightarrow{E}$ be defined as follows:

$$\overrightarrow{E} = \left(\bigvee_{i=1}^{n} l_i \text{ where } l_i \text{ is in } \overrightarrow{C} \text{ or in } \overrightarrow{D} \text{ and } l_i\theta \text{ is the } i\text{-th literal of } \overrightarrow{\bot} \right)\sigma.$$

We assume that the disjunction notion with indexes from $i = 1$ to n used to define $\overrightarrow{E}$ means that the literals l_i of $\overrightarrow{E}$ follow the same order as literals in $\overrightarrow{\bot}$. We show that $\overrightarrow{E}$ is in $\overrightarrow{\mathcal{L}}_\bot$ and it is a mgs$_\bot$ for $\overrightarrow{C}$ and $\overrightarrow{D}$. Let h_1 and h_2 be the heads of $\overrightarrow{C}$ and $\overrightarrow{D}$ respectively. h_1 and h_2 are among compatible pair of literals in S and they are unified by σ and therefore $\overrightarrow{E}$ is a definite ordered clause. Moreover, by definition $\overrightarrow{E}\theta$ is derived from $\overrightarrow{\bot}$ by removing some literals without changing the order of the remaining literals. Then according to Lemma 5, $\overrightarrow{E}\theta$ is a subsequence of $\overrightarrow{\bot}$ and therefore $\overrightarrow{E}$ is in $\overrightarrow{\mathcal{L}}_\bot$. We have $\overrightarrow{C} \succeq_\bot \overrightarrow{E}$ and $\overrightarrow{D} \succeq_\bot \overrightarrow{E}$, since by definition $\overrightarrow{C}\sigma \sqsubseteq_\bot \overrightarrow{E}$ and $\overrightarrow{D}\sigma \sqsubseteq_\bot \overrightarrow{E}$. Suppose $\overrightarrow{F}$ is a clause in $\overrightarrow{\mathcal{L}}_\bot$ such that $\overrightarrow{C} \succeq_\bot \overrightarrow{F}$ and $\overrightarrow{D} \succeq_\bot \overrightarrow{F}$. In order to establish that $\overrightarrow{E}$ is an mgs$_\bot$ of $\overrightarrow{C}$ and $\overrightarrow{D}$, we need to prove $\overrightarrow{E} \succeq_\bot \overrightarrow{F}$. Let θ_1' and θ_2' be variable substitutions such that $\overrightarrow{C}\theta_1' \sqsubseteq_\bot \overrightarrow{F}$ and $\overrightarrow{D}\theta_2' \sqsubseteq_\bot \overrightarrow{F}$, h' be the head of $\overrightarrow{F}$ and $\theta' = \{\theta_1' \cup \theta_2'\}$. Then, $h_1\theta' = h_1\theta_1' = h'$ and $h_2\theta' = h_2\theta_2' = h'$, so θ' is a unifier for h_1 and h_2. But σ is an *mgu* for h_1 and h_2 and so there is a substitution γ such that $\theta' = \sigma\gamma$. According to the definition of $\overrightarrow{E}$, for every literal $l_i\sigma$ in $\overrightarrow{E}$, l_i is either in $\overrightarrow{C}$ or in $\overrightarrow{D}$. But $\overrightarrow{C}\theta_1 \sqsubseteq_\bot \overrightarrow{F}$ and $\overrightarrow{D}\theta' \sqsubseteq_\bot \overrightarrow{F}$ and therefore each literal $(l_i\sigma)\gamma$ in $\overrightarrow{E}\gamma$ is mapped to an equivalent literal $l_i\theta'$ in $\overrightarrow{F}$ and both $\overrightarrow{E}$ and $\overrightarrow{F}$ are subsequences of $\overrightarrow{\bot}$. Then according to Lemma 5, $\overrightarrow{E}\gamma$ is a subsequence of $\overrightarrow{F}$ relative to $\bot$ and we have $\overrightarrow{E} \succeq_\bot \overrightarrow{F}$. $\overrightarrow{E}$ is therefore an mgs$_\bot$ for $\overrightarrow{C}$ and $\overrightarrow{D}$ in $\overrightarrow{\mathcal{L}}_\bot$. □

Example 12 Let $\overrightarrow{C}$, $\overrightarrow{D}$ and $\overrightarrow{\bot}$ be ordered clauses as defined below:

$$\begin{aligned}
\overrightarrow{C} &= \mathrm{p}(X_1, Y_1) \leftarrow \mathrm{q}(X_1, X_1), \mathrm{q}(Y_1, W_1),\\
\overrightarrow{D} &= \mathrm{p}(X_2, Y_2) \leftarrow \mathrm{q}(Z_2, X_2), \mathrm{q}(Y_2, Y_2), \mathrm{r}(X_2, Y_2),\\
\overrightarrow{\bot} &= \mathrm{p}(X, Y) \leftarrow \mathrm{q}(X, X), \mathrm{q}(Y, Y), \mathrm{r}(X, Y), \mathrm{s}(X, Y).
\end{aligned}$$

Suppose that $\overrightarrow{C}$ and $\overrightarrow{D}$ are in $\overrightarrow{\mathcal{L}}_\bot$ and literals of $\overrightarrow{C}$ are respectively mapped to the first, second and third literals of $\overrightarrow{\bot}$ and literals of $\overrightarrow{D}$ are respectively mapped to the first, second, third and fourth literals of $\overrightarrow{\bot}$. Then according to Definition 24, S, $\overrightarrow{C_S}$ and $\overrightarrow{D_S}$ are defined as follows:

$$\begin{aligned}
S = {} & (\mathrm{p}(X_1, Y_1), \mathrm{p}(X_2, Y_2)), (\neg\mathrm{q}(X_1, X_1), \neg\mathrm{q}(Z_2, X_2)),\\
& (\neg\mathrm{q}(Y_1, W_1), \neg\mathrm{q}(Y_2, Y_2)),\\
\overrightarrow{C_S} = {} & \mathrm{p}(X_1, Y_1) \leftarrow \mathrm{q}(X_1, X_1), \mathrm{q}(Y_1, W_1),\\
\overrightarrow{D_S} = {} & \mathrm{p}(X_2, Y_2) \leftarrow \mathrm{q}(Z_2, X_2), \mathrm{q}(Y_2, Y_2).
\end{aligned}$$

$\overrightarrow{C}$ and $\overrightarrow{D}$ are in $\overrightarrow{\mathcal{L}}_\bot$ and there are substitutions $\theta_1 = \{X_1/X,\ Y_1/Y,\ W_1/Y\}$ and $\theta_2 = \{X_2/X, Y_2/Y, Z_2/X\}$ such that $\overrightarrow{C}\theta_1$ and $\overrightarrow{D}\theta_2$ are subsequences of $\overrightarrow{\bot}$. Let $\sigma = \{X_2/X_1, Y_2/Y_1, Z_2/X_1, W_1/Y_1\}$ be an *mgu* for $\{\overrightarrow{C_S}, \overrightarrow{D_S}\}$, then according to Theorem 4, mgs$_\bot$ for $\overrightarrow{C}$ and $\overrightarrow{D}$ is defined as follows:

$$\begin{aligned}
\overrightarrow{E} &= (\mathrm{p}(X_1, Y_1) \leftarrow \mathrm{q}(X_1, X_1), \mathrm{q}(Y_1, W_1), \mathrm{r}(X_2, Y_2))\sigma\\
&= \mathrm{p}(X_1, Y_1) \leftarrow \mathrm{q}(X_1, X_1), \mathrm{q}(Y_1, Y_1), \mathrm{r}(X_1, Y_1).
\end{aligned}$$

In the following we prove the existence of lgg$_\bot$ in $\overrightarrow{\mathcal{L}}_\bot$. We use the lgg$_a$ for atoms as a bridge to find lgg$_\bot$. This is similar to the proof for the existence of lgg for conventional clauses (Nienhuys-Cheng and de Wolf 1997) adopted for ordered clauses and subsumption relative to a bottom clause.

Definition 25 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. If $n = m$ and for every $i = 1, \ldots, n$, L_i and M_i have the same sign and predicate symbol, we say C and D are compatible clauses. $\overrightarrow{C}$ is an atomic generalisation of $\overrightarrow{D}$, denoted by $\overrightarrow{C} \succeq_a \overrightarrow{D}$, if $\overrightarrow{C}$ and $\overrightarrow{D}$ are compatible clauses and there exists a substitution θ such that $L_i\theta = M_i$ for every $i = 1, \ldots, n$.

In this paper we use the same notion used in Nienhuys-Cheng and de Wolf (1997) for atomic representation of ordered clauses.

Definition 26 (Atomic representation) Let $\overrightarrow{C} = L_1 \vee L_2 \vee \cdots \vee L_n$ be an ordered clause. Atomic representation of clause $\overrightarrow{C}$ is denoted by $a(\overrightarrow{C}) = \vee(L_1, L_2, \ldots, L_n)$ where $\vee$ acts as a n-ary predicate symbol and $L_1, L_2, \ldots, L_n$ as terms.

In this definition we assume an appropriate mapping between predicate symbols in $\overrightarrow{C}$ and function symbols in $a(\overrightarrow{C})$.

Theorem 5 *For every compatible ordered clauses* $\overrightarrow{C} = L_1 \vee L_2 \vee \cdots \vee L_n$ *and* $\overrightarrow{D} = M_1 \vee M_2 \vee \cdots \vee M_n$ *in* $\overrightarrow{\mathcal{L}}_\perp$, *there exists an* $\mathrm{lgg}_a(\overrightarrow{C}, \overrightarrow{D})$ *in* $\overrightarrow{\mathcal{L}}_\perp$.

Proof Let $A = \vee(L_1, \ldots, L_n)$ and $B = \vee(M_1, \ldots, M_n)$ be the atomic representations of $\overrightarrow{C}$ and $\overrightarrow{D}$. Let $\vee(N_1, \ldots, N_n)$ be the least general generalisation (lgg) of atoms A and B obtained from the anti-unification algorithm (e.g. Algorithm 13.1 in Nienhuys-Cheng and de Wolf 1997). We need to show that $\overrightarrow{E} = N_1 \vee N_2 \vee \cdots \vee N_n$ is an $\mathrm{lgg}_a(\overrightarrow{C}, \overrightarrow{D})$ and that $\overrightarrow{E}$ is in $\overrightarrow{\mathcal{L}}_\perp$. $\overrightarrow{C}$, $\overrightarrow{D}$ and $\overrightarrow{E}$ are compatible ordered clauses and they can be viewed as atoms and therefore the proof that $\overrightarrow{E}$ is an $\mathrm{lgg}_a(\overrightarrow{C}, \overrightarrow{D})$ follows from the proof of this theorem for atoms (e.g. Proposition 13.23 in Nienhuys-Cheng and de Wolf 1997). We show that $\overrightarrow{E}$ is in $\overrightarrow{\mathcal{L}}_\perp$. According to definition of lgg_a we have $\overrightarrow{E} \succeq_a \overrightarrow{C}$ and $\overrightarrow{E} \succeq_a \overrightarrow{D}$ and therefore there exist variable substitutions θ_1 and θ_2 such that $\overrightarrow{E}\theta_1 = \overrightarrow{C}$ and $\overrightarrow{E}\theta_2 = \overrightarrow{D}$. But $\overrightarrow{C}$ and $\overrightarrow{D}$ are in $\overrightarrow{\mathcal{L}}_\perp$ and therefore $\overrightarrow{E}$ is in $\overrightarrow{\mathcal{L}}_\perp$. □

Theorem 6 (Existence of $\mathrm{lgg}_\perp$ in $\overrightarrow{\mathcal{L}}_\perp$) *For every ordered clauses* $\overrightarrow{C}$ *and* $\overrightarrow{D}$ *in* $\overrightarrow{\mathcal{L}}_\perp$, *there exists an* $\mathrm{lgg}_\perp$ *of* $\overrightarrow{C}$ *and* $\overrightarrow{D}$ *in* $\overrightarrow{\mathcal{L}}_\perp$.

Proof Let S, $\overrightarrow{C_S}$ and $\overrightarrow{D_S}$ be as defined in Definition 24 such that S is a sequence of all pairs of compatible literals from $\overrightarrow{C}$ and $\overrightarrow{D}$ relative to $\perp$. $\overrightarrow{C_S}$ and $\overrightarrow{D_S}$ are compatible clauses in $\overrightarrow{\mathcal{L}}_\perp$ and according to Theorem 5 there is an lgg_a for $\overrightarrow{C_S}$ and $\overrightarrow{D_S}$ in $\overrightarrow{\mathcal{L}}_\perp$. Let $\overrightarrow{E} = \mathrm{lgg}_a(\overrightarrow{C_S}, \overrightarrow{D_S})$. $\overrightarrow{E}$ is in $\overrightarrow{\mathcal{L}}_\perp$ and we show that it is a $\mathrm{lgg}_\perp$ for $\overrightarrow{C}$ and $\overrightarrow{D}$. According to definition of lgg_a we have $\overrightarrow{E} \succeq_a \overrightarrow{C_S}$ and $\overrightarrow{E} \succeq_a \overrightarrow{D_S}$, but according to Definition 24, $\overrightarrow{C_S} \succeq_\perp \overrightarrow{C}$ and $\overrightarrow{D_S} \succeq_\perp \overrightarrow{D}$. By transitivity of $\succeq_\perp$ we have $\overrightarrow{E} \succeq_\perp \overrightarrow{C}$ and $\overrightarrow{E} \succeq_\perp \overrightarrow{D}$. Let $\overrightarrow{F} = N_1 \vee N_2 \vee \cdots \vee N_m$ be a clause in $\overrightarrow{\mathcal{L}}_\perp$ such that $\overrightarrow{F} \succeq_\perp \overrightarrow{C}$ and $\overrightarrow{F} \succeq_\perp \overrightarrow{D}$. In order to establish that $\overrightarrow{E}$ is an $\mathrm{lgg}_\perp$ of $\overrightarrow{C}$ and $\overrightarrow{D}$, we need to prove $\overrightarrow{F} \succeq_\perp \overrightarrow{E}$. Since $\overrightarrow{F} \succeq_\perp \overrightarrow{C}$ and $\overrightarrow{F} \succeq_\perp \overrightarrow{D}$, there are variable substitutions θ_1' and θ_2' and literals $L_1 \vee \cdots \vee L_m \sqsubseteq_\perp \overrightarrow{C}$ and $M_1 \vee \cdots \vee M_m \sqsubseteq_\perp \overrightarrow{D}$, such that $N_i\theta_1' = L_i$ and $N_i\theta_2' = M_i$, for every $1 \le i \le m$. Then $S' = (L_1, M_1), \ldots, (L_m, M_m)$ is a sequence of pairs of compatible literals from $\overrightarrow{C}$ and $\overrightarrow{D}$ relative to $\perp$. Let $\overrightarrow{C_{S'}} = L_1 \vee L_2 \vee \cdots \vee L_m$ and $\overrightarrow{D_{S'}} = M_1 \vee M_2 \vee \cdots \vee M_m$ as defined in Definition 24, let $\overrightarrow{G} = K_1 \vee K_2 \vee \cdots \vee K_m$ be an $\mathrm{lgg}_a(\overrightarrow{C_{S'}}, \overrightarrow{D_{S'}})$, and σ_1 and σ_2 be such that $\overrightarrow{G}\sigma_1 = C_{S'}$ and $\overrightarrow{G}\sigma_2 = D_{S'}$. Since $(N_1 \vee N_2 \vee \cdots \vee N_m)\theta_1' = C_{S'}$ and $(N_1 \vee N_2 \vee \cdots \vee N_m)\theta_2' = D_{S'}$, there must be a γ such that $(N_1 \vee N_2 \vee \cdots \vee N_m)\gamma = K_1 \vee K_2 \vee \cdots \vee K_m$. We have $(N_1 \vee N_2 \vee \cdots \vee N_m)\gamma = \overrightarrow{G}$, so $\overrightarrow{F} \succeq_\perp \overrightarrow{G}$. Since every pair of literals in S' is also in S, there is a substitution γ' such that each literal in $\overrightarrow{F}\gamma'$ is also in $\overrightarrow{G}$. Then according to Lemma 5 we have $\overrightarrow{G} \succeq_\perp \overrightarrow{E}$. Hence, $\overrightarrow{F} \succeq_\perp \overrightarrow{E}$ by transitivity of $\succeq_\perp$. □

Thus the $\text{lgg}_\perp$ of any pair of ordered clauses in $\overrightarrow{\mathcal{L}}_\perp$ exists and can be computed by the following algorithm:

Algorithm 1 ($\text{lgg}_\perp(\overrightarrow{C}, \overrightarrow{D})$)

```
1. Given two ordered clauses $\overrightarrow{C}$ and $\overrightarrow{D}$ in $\overrightarrow{\mathcal{L}}_\perp$.
2. Let $S$, $\overrightarrow{C_S}$ and $\overrightarrow{D_S}$ be as defined in Definition 24 such that $S$ is a sequence of all pairs of
   compatible literals from $\overrightarrow{C}$ and $\overrightarrow{D}$ relative to $\perp$.
3. Obtain $\overrightarrow{E} = \text{lgg}_a(\overrightarrow{C_S}, \overrightarrow{D_S})$.
4. Return $\overrightarrow{E}$
```

Note that $\overrightarrow{C}$ and $\overrightarrow{D}$ are definite ordered clauses in $\overrightarrow{\mathcal{L}}_\perp$ with the same predicate symbol in the head and there is at least one pair of compatible literals from $\overrightarrow{C}$ and $\overrightarrow{D}$ relative to $\perp$ which is the pair of heads of $\overrightarrow{C}$ and $\overrightarrow{D}$. Hence, $\text{lgg}_\perp$ for $\overrightarrow{C}$ and $\overrightarrow{D}$ has at least one literal. Moreover, each literal from $\overrightarrow{C}$ and $\overrightarrow{D}$ can only be mapped to one literal from $\overrightarrow{\perp}$ and therefore each literal in $\overrightarrow{C}$ can be mapped to at most one literal from $\overrightarrow{D}$. Thus, $\overrightarrow{C}$ and $\overrightarrow{D}$ can have at most $\min(|\overrightarrow{C}|, |\overrightarrow{D}|)$ pairs of compatible literals with respect to $\perp$ and accordingly $\text{lgg}_\perp(\overrightarrow{C}, \overrightarrow{D})$ has at most $\min(|\overrightarrow{C}|, |\overrightarrow{D}|)$ literals. Note that the lgg of $\overrightarrow{C}$ and $\overrightarrow{D}$ with respect to the general subsumption order has at most $|\overrightarrow{C}| \times |\overrightarrow{D}|$ literals as $\overrightarrow{C}$ and $\overrightarrow{D}$ can have at most $|\overrightarrow{C}| \times |\overrightarrow{D}|$ pairs of compatible literals.

Example 13 Let $\overrightarrow{C}$, $\overrightarrow{D}$, $\overrightarrow{\perp}$, S, $\overrightarrow{C_S}$ and $\overrightarrow{D_S}$ be as defined in Example 12. Then according to Theorem 6, $\text{lgg}_\perp$ for $\overrightarrow{C}$ and $\overrightarrow{D}$ is defined as follows:

$$\begin{aligned}\overrightarrow{E} &= \text{lgg}_a(\overrightarrow{C_S}, \overrightarrow{D_S})\\ &= \text{p}(X, Y) \leftarrow \text{q}(Z, X), \text{q}(Y, W)\end{aligned}$$

Since we have proved the existence of $\text{mgs}_\perp$ and $\text{lgg}_\perp$ of any pair of ordered clauses in $\overrightarrow{\mathcal{L}}_\perp$, it follows that $\overrightarrow{\mathcal{L}}_\perp$ ordered by subsumption relative to a bottom clause has a lattice structure.

Theorem 7 $\langle \overrightarrow{\mathcal{L}}_\perp, \succeq_\perp \rangle$ *is a lattice.*

Proof According to Theorem 3 $\langle \overrightarrow{\mathcal{L}}_\perp, \succeq_\perp \rangle$ is a quasi-order. According to Theorems 4 and 6 each pair of ordered clauses in $\overrightarrow{\mathcal{L}}_\perp$ have a most general specialisation ($\text{mgs}_\perp$) and a least general generalisation ($\text{lgg}_\perp$) in $\overrightarrow{\mathcal{L}}_\perp$. Hence, $\langle \overrightarrow{\mathcal{L}}_\perp, \succeq_\perp \rangle$ is a lattice. □

In the following, we show the morphism between the lattice $\langle \overrightarrow{\mathcal{L}}_\perp, \succeq_\perp \rangle$ and an atomic lattice. First we define the set of atoms which are generalisations of the atomic representation of $\overrightarrow{\perp}$. In this definition we use the same notion used in Nienhuys-Cheng and de Wolf (1997) for atomic representation of ordered clauses as defined in Definition 26.

Definition 27 ($\mathcal{A}_\perp$) Let $\overrightarrow{\perp}$ be the bottom clause as defined in Definition 11, $a(\overrightarrow{\perp})$ be the atomic representation of $\overrightarrow{\perp}$ as defined in Definition 26 and A be an atom. A is in $\mathcal{A}_\perp$ if and only if there exists a substitution θ such that $A\theta = a(\overrightarrow{\perp})$.

Definition 28 (Mapping function f) Let $\mathcal{A}_\perp$ and $\overrightarrow{\mathcal{L}}_\perp$ be as defined in Definition 27 and Definition 20 and A be an atom. The mapping function $f : \mathcal{A}_\perp \to \overrightarrow{\mathcal{L}}_\perp$ is defined as follows:

$$f(A) = \left(\bigvee_{i=1}^{n} l_i \text{ where } l_i \text{ is the } i\text{-th term from } A \text{ which is not a variable}\right)$$

Note that in Definition 28 a non-variable term in A represents a literal in $f(A)$ and a variable in A represents the absence of a literal from $\overrightarrow{\bot}$ in $f(A)$. This variable is always a distinct variable because, according to Definition 27, it can only be substituted by a distinct non-variable term from $a(\overrightarrow{\bot})$. Hence, in Definition 28, if l_i is a variable then it is always distinct and cannot be unified with other variables in A. In order to simplify the representation we replace such a variable by the symbol '_'.

Example 14 Let $\overrightarrow{\bot}$ be the bottom clause as in Fig. 4. Then the atomic representation of $\overrightarrow{\bot}$ (i.e. $a(\overrightarrow{\bot})$) and atoms A_1 and A_2 in $\mathcal{A}_\perp$ are as follows:

$$\begin{aligned} A_1 &= \vee\big(p(X),\ \_,\ \neg r(X),\ \_,\ \neg s(Y,Z)\big),\\ A_2 &= \vee\big(p(X),\ \_,\ \neg r(X),\ \_,\ \neg s(Y,X)\big),\\ a(\overrightarrow{\bot}) &= \vee\big(p(X),\ \neg q(X),\ \neg r(X),\ \neg s(X,Y),\ \neg s(Y,X)\big). \end{aligned}$$

Atoms A_1 and A_2 correspond to ordered clauses $\overrightarrow{C}$ and $\overrightarrow{D}$ in Fig. 4 as $\overrightarrow{C} = f(A_1)$ and $\overrightarrow{D} = f(A_2)$.

Theorem 8 *Let $\mathcal{A}_\perp$ be as defined in Definition 27 and the mapping function f as defined in Definition 28. Let A and B be atoms in $\mathcal{A}_\perp$, then we have $f(A) \succeq_\perp f(B)$ if and only if $A \succeq B$.*

Proof Let $A = \vee(L_1, L_2, \dots, L_n)$, $B = \vee(M_1, M_2, \dots, M_n)$ and $a(\overrightarrow{\bot}) = \vee(N_1, N_2, \dots, N_n)$. Let $\overrightarrow{C}$ and $\overrightarrow{D}$ be ordered clauses such that $\overrightarrow{C} = f(A)$ and $\overrightarrow{D} = f(B)$.

$\Rightarrow$: Suppose $\overrightarrow{C} \succeq_\perp \overrightarrow{D}$, then there exists a substitution θ such that $\overrightarrow{C}\theta$ is a subsequence of $\overrightarrow{D}$ relative to $\perp$. Then according to Definition 21 for each literal in $\overrightarrow{C}\theta$ there is an equivalent literal in $\overrightarrow{D}$ and both literals correspond to the same literal from $\overrightarrow{\bot}$. Hence, for each term L_i in A if L_i is not a variable then it corresponds to a literal in $\overrightarrow{C}$ and therefore there exist a non-variable term M_i in B such that $L_i\theta = M_i$. If L_i is a variable then there exist a substitution θ' such that $L_i\theta' = M_i$ where $L_i/M_i \in \theta'$ and M_i is a variable or non-variable term in B. Hence, for each term L_i in A there is a substitution $\sigma = \theta \cup \theta'$ such that $L_i\sigma = M_i$. This implies $A\sigma = B$ and therefore $A \succeq B$.

$\Leftarrow$: Suppose $A \succeq B$, then there exists a substitution θ such that $A\theta = B$. Then for each L_i in A we have term M_i in B. Moreover, both terms L_i and M_i correspond to the same term N_i from $a(\overrightarrow{\bot})$. Hence, for each literal in $\overrightarrow{C}\theta$ there is an equivalent literal in $\overrightarrow{D}$, both literals correspond to the same literal from $\overrightarrow{\bot}$ and both $\overrightarrow{C}\theta$ and $\overrightarrow{D}$ are subsequences of $\overrightarrow{\bot}$. Then according to Lemma 5, $\overrightarrow{C}\theta$ is a subsequence of $\overrightarrow{D}$ relative to $\perp$ and we have $\overrightarrow{C} \succeq_\perp \overrightarrow{D}$. □

Example 15 Let $\overrightarrow{\bot}$ be the bottom clause and $\overrightarrow{C}$, $\overrightarrow{D}$ and $\overrightarrow{E}$ be the ordered clauses as in Fig. 4. Atoms A_1, A_2 and A_3 in Fig. 5 correspond to ordered clauses $\overrightarrow{C}$, $\overrightarrow{D}$ and $\overrightarrow{E}$ and we have $\overrightarrow{C} = f(A_1)$, $\overrightarrow{D} = f(A_2)$ and $\overrightarrow{E} = f(A_3)$. We have $\overrightarrow{C} \succeq_\perp \overrightarrow{D}$ and $A_1 \succeq A_2$ as shown in Fig. 5.a and $\overrightarrow{C} \not\succeq_\perp \overrightarrow{E}$ and $A_1 \not\succeq A_3$ as shown in Fig. 5.b.

$$\begin{tikzcd}
A_1 = & \vee(p(X), \arrow[d] & \_, & \neg r(X), \arrow[d] & \_, & \neg s(Y,Z)) \arrow[d] \\
A_2 = & \vee(p(X), \arrow[d, dotted] & \_, & \neg r(X), \arrow[d, dotted] & \_, & \neg s(Y,X)) \arrow[d, dotted] \\
a(\overrightarrow{\bot}) = & \vee(p(X), & \neg q(X), & \neg r(X), & \neg s(X,Y), & \neg s(Y,X))
\end{tikzcd}$$

(a)

$$\begin{tikzcd}
A_1 = & \vee(p(X), \arrow[d] & \_, & \neg r(X), \arrow[d] & \_, & \neg s(Y,Z)) \arrow[d, dotted, no head] \\
A_3 = & \vee(p(X), \arrow[d, dotted] & \_, & \neg r(X), \arrow[d, dotted] & \neg s(X,Y), \arrow[d, dotted] & \_) \arrow[d, dotted] \\
a(\overrightarrow{\bot}) = & \vee(p(X), & \neg q(X), & \neg r(X), & \neg s(X,Y), & \neg s(Y,X))
\end{tikzcd}$$

(b)

Fig. 5 Subsumption relative to a bottom clause can be mapped to the atomic subsumption. Atoms A_1, A_2 and A_3 correspond to ordered clauses $\overrightarrow{C}$, $\overrightarrow{D}$ and $\overrightarrow{E}$ in Fig. 4 and we have $\overrightarrow{C} = f(A_1)$, $\overrightarrow{D} = f(A_2)$ and $\overrightarrow{E} = f(A_3)$. (**a**) $\overrightarrow{C} \succeq_\bot \overrightarrow{D}$ and $A_1 \succeq A_2$ (**b**) $\overrightarrow{C} \not\succeq_\bot \overrightarrow{E}$ and $A_1 \not\succeq A_3$

Theorem 9 *The mapping function $f : \mathcal{A}_\bot \to \overrightarrow{\mathcal{L}}_\bot$ as defined in Definition 28 is an order-isomorphism.*

Proof First we show that the mapping function f is onto. Let $\overrightarrow{C}$ be an ordered clause in $\overrightarrow{\mathcal{L}}_\bot$, then according to Definitions 20 and 21, there exist substitution θ such that $\overrightarrow{C}\theta \sqsubseteq_\bot \overrightarrow{\bot}$ and therefore $\overrightarrow{C} \succeq_\bot \overrightarrow{\bot}$. Then, according to Theorem 8 we have $A \succeq a(\overrightarrow{\bot})$ where $\overrightarrow{C} = f(A)$ and therefore according to Definition 27, A is in $\mathcal{A}_\bot$. Hence, the mapping function f is onto. Moreover, according to Theorem 8, the mapping function f is an order-embedding. Then according to Definition 2, f is an order-isomorphism. □

We have shown that $\langle \overrightarrow{\mathcal{L}}_\bot, \succeq_\bot \rangle$ is a lattice. It is also known that the atomic subsumption defines a lattice (Reynolds 1969). The proposition below follows directly from Theorem 9 and Remark 4.

Proposition 1 *The mapping function $f : \mathcal{A}_\bot \to \overrightarrow{\mathcal{L}}_\bot$ as defined in Definition 28 is a lattice isomorphism and lattices $\langle \overrightarrow{\mathcal{L}}_\bot, \succeq_\bot \rangle$ and $\langle \mathcal{A}_\bot, \succeq \rangle$ are two isomorphic lattices.*

The proposition below follows from f being a lattice isomorphism.

Proposition 2 *Let $\mathcal{A}_\bot$ and mapping function f be defined as in Definition 28 and A and B be atoms in $\mathcal{A}_\bot$. The mapping function $f : \mathcal{A}_\bot \to \overrightarrow{\mathcal{L}}_\bot$ is join-preserving and meet-preserving that is*:

1. $\mathrm{mgs}_\bot(f(A), f(B)) = f(\mathrm{mgs}(A, B))$,
2. $\mathrm{lgg}_\bot(f(A), f(B)) = f(\mathrm{lgg}(A, B))$.

Example 16 Let $a(\overrightarrow{\bot})$ and atoms A and B in $\mathcal{A}_\bot$ be as defined below:

$$A = \vee\big(p(X_1, Y_1), \neg q(X_1, X_1), \neg q(Y_1, W_1), \_, \_\big),$$

$$B = \vee\big(p(X_2, Y_2), \neg q(Z_2, X_2), \neg q(Y_2, Y_2), \neg r(X_2, Y_2), \_\big),$$

$$a(\overrightarrow{\bot}) = \vee\big(p(X, Y), \neg q(X, X), \neg q(Y, Y), \neg r(X, Y), \neg s(X, Y)\big).$$

According to Definition 28, $f(A)$ and $f(B)$ are defined as follows:

$$\overrightarrow{C} = f(A) = p(X_1, Y_1) \leftarrow q(X_1, X_1), q(Y_1, W_1),$$

$$\overrightarrow{D} = f(B) = p(X_2, Y_2) \leftarrow q(Z_2, X_2), q(Y_2, Y_2), r(X_2, Y_2).$$

mgs(A, B) and lgg(A, B) are defined as follows:

$$\mathrm{mgs}(A, B) = \vee\big(p(X_1, Y_1), \neg q(X_1, X_1), \neg q(Y_1, Y_1), \neg r(X_1, Y_1), \_\big),$$

$$\mathrm{lgg}(A, B) = \vee\big(p(X_1, Y_1), \neg q(Z_1, X_1), \neg q(Y_1, W_1), \_, \_\big).$$

Then according to Proposition 1, mgs$_\bot$ and lgg$_\bot$ for $\overrightarrow{C}$ and $\overrightarrow{D}$ are defined as follows:

$$\mathrm{mgs}_\bot(\overrightarrow{C}, \overrightarrow{D}) = p(X_1, Y_1) \leftarrow q(X_1, X_1), q(Y_1, Y_1), r(X_1, Y_1),$$

$$\mathrm{lgg}_\bot(\overrightarrow{C}, \overrightarrow{D}) = p(X_1, Y_1) \leftarrow q(Z_1, X_1), q(Y_1, W_1).$$

As previously shown in this section, for ordered clauses $\overrightarrow{C}$ and $\overrightarrow{D}$ in $\overrightarrow{\mathcal{L}}_\bot$ we have at most $\min(|\overrightarrow{C}|, |\overrightarrow{D}|)$ pairs of compatible literals relative to $\bot$ and accordingly lgg$_\bot(\overrightarrow{C}, \overrightarrow{D})$ has at most $\min(|\overrightarrow{C}|, |\overrightarrow{D}|)$ literals. Similarly, Proposition 1 suggest that lgg$_\bot(\overrightarrow{C}, \overrightarrow{D})$ has at most $|\overrightarrow{\bot}|$ literals as $|\overrightarrow{C}|$ and $|\overrightarrow{D}|$ are bounded by $|\overrightarrow{\bot}|$.

It is known that general subsumption testing is NP-complete (Garey and Johnson 1979). A subsumption relation for ordered clauses (i.e. ordered subsumption) is studied in Kuwabara et al. (2006). It is shown that ordered subsumption testing is NP-complete. By a similar proof it can be shown that an instance of 3-SAT problem can be reduced to the sequential subsumption testing and therefore sequential subsumption testing is also NP-complete. However, as shown in this section, subsumption testing relative to a bottom clause can be mapped to atomic subsumption testing which is polynomial time.

We have shown that the subsumption order relative to a bottom clause defines a lattice and this lattice is isomorphic to an atomic lattice. The complexity of the hypothesis space bounded by a bottom clause can be further analysed by mapping the lattice $\langle \overrightarrow{\mathcal{L}}_\bot, \succeq_\bot \rangle$ to a partition lattice. This analysis is not given in this paper. However, the morphism between the function free atomic lattice and the lattice of partitions is given in Appendix B.

7 Ideal refinement operators for the subsumption order relative to $\bot$

It is known that when a full Horn clause language and the general subsumption order are considered, there exist no ideal refinement operators (van der Laag and Nienhuys-Cheng 1994). However, if $\langle \mathcal{L}, \geq \rangle$ is a quasi-order, $\mathcal{L}$ is finite and $\geq$ is decidable, then there exists an ideal refinement operator for $\langle \mathcal{L}, \geq \rangle$ (Nienhuys-Cheng and de Wolf 1997). Hence, given the finiteness of $\overrightarrow{\mathcal{L}}_\bot$ one could expect the existence of ideal refinement operators for $\langle \overrightarrow{\mathcal{L}}_\bot, \succeq_\bot \rangle$. In this section we define a refinement operator ρ_1 and show that ρ_1 is ideal for $\langle \overrightarrow{\mathcal{L}}_\bot, \succeq_\bot \rangle$. First we define a mapping function which is used in the refinement operator. According to Definition 20, for each ordered clause $\overrightarrow{C}$ in $\overrightarrow{\mathcal{L}}_\bot$ there exists a substitution θ such that $\overrightarrow{C}\theta$ is a subsequence of $\overrightarrow{\bot}$. Thus, there exists a selection function s which maps each literal of $\overrightarrow{C}\theta$ to a literal of $\overrightarrow{\bot}$ 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 $\overrightarrow{\bot}$. 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 order to decode the clause from $\overrightarrow{\bot}$. In this setting, substitution θ maps variables from $\overrightarrow{C}$ to the variables of $\overrightarrow{\bot}$. The decoding, therefore, requires inverse substitution θ^{-1}. This can be achieved by maintaining the position of variables when the substitution θ is constructed (Nienhuys-Cheng and de Wolf 1997). 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 $\overrightarrow{\bot}$ with all variables replaced with new and distinct variables.[3]

Definition 29 (Mapping function c) Let $\overrightarrow{\bot}$ and $\overrightarrow{\mathcal{L}}_{\bot}$ be as defined in Definition 20, 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 v_i and v_j be distinct variables in $\overrightarrow{\top}$ such that $i < j$. Let θ be a variable substitution in Θ, where $\Theta = \{\theta | \theta \subseteq \hat{\theta}_{\top}$ and if $\{v_j/u, u/v_i\} \subseteq \theta$ then $v_j/v_i \in \theta\}$ and $\hat{\theta}_{\top} = \{v_j/v_i | \{v_i/u, v_j/u\} \subseteq \theta_{\top}\}$. Let $\mathcal{K}$ be power set of $\{1, \ldots, n\}$. The mapping function $c : \mathcal{K} \times \Theta \to \overrightarrow{\mathcal{L}}_{\bot}$ is defined as follows:

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

In this definition $\theta_{\top}$ is a substitution which maps variables in $\overrightarrow{\top}$ to variables in $\overrightarrow{\bot}$, $\hat{\theta}_{\top}$ is a substitution containing valid bindings between the variables in $\overrightarrow{\top}$ with respect to $\overrightarrow{\bot}$ and Θ is the set of all subsets of $\hat{\theta}_{\top}$ containing transitive bindings. Note that the disjunction notion with indexes from $i = 1$ to n used in Definition 29 means that the literals l_i of $c(\langle K, \theta\rangle)$ follow the same order as literals in $\overrightarrow{\top}$.

Example 17 Let $\overrightarrow{\bot}$ be the bottom clause in Example 3. $\overrightarrow{\top}$ can be obtained from $\overrightarrow{\bot}$ by replacing all variables with new and distinct variables:

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

Let $K = \{1, 2, 5\}$, $\theta = \{V_4/V_1, V_{12}/V_5\}$ then $c(\langle K, \theta\rangle)$ can be defined as follows:

$$\overrightarrow{C} = c(\langle K, \theta\rangle) = \text{mult}(V_1, V_2, V_3) \leftarrow \text{dec}(V_1, V_5), \text{mult}(V_5, V_{13}, V_{14}).$$

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 is also used to make 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 (Laird 1987) adopted for subsumption relative to $\bot$ and a refinement space bounded below by a bottom clause.

Definition 30 (ρ_1) Let $\overrightarrow{\bot}$ and $\overrightarrow{\mathcal{L}}_{\bot}$ be as defined in Definition 20, $\overrightarrow{C}$ be an ordered clause in $\overrightarrow{\mathcal{L}}_{\bot}$, n be the number of literals in $\bot$, k be a natural number, $1 \le k \le n$, $\overrightarrow{\top}$, Θ and $\mathcal{K}$ be

[3] $\overrightarrow{\top}$ should not be confused with the top element in $\overrightarrow{\mathcal{L}}_{\bot}$ which is the empty clause $\Box$.

defined as in Definition 29. 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 29. $\langle \overrightarrow{C'}, K', \theta'\rangle$ 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 30, ρ_1 adds a most general literal from $\overrightarrow{\top}$ which has not been added before (choice 1) or it applies an elementary variable substitution such that the clause subsumes $\overrightarrow{\bot}$ (choice 2).

Example 18 Let $\overrightarrow{\top}$, K, θ and $\overrightarrow{C}$ be as defined in Example 17. Then $\langle \overrightarrow{C'}, K', \theta'\rangle$ is in $\rho_1(\langle \overrightarrow{C}, K, \theta\rangle)$ and (i) $K' = \{1, 2, 5, 6\}$, $\theta' = \{V_4/V_1, V_{12}/V_5\}$ and

$$\overrightarrow{C'} = c(\langle K', \theta'\rangle) = \text{mult}(V_1, V_2, V_3) \leftarrow \text{dec}(V_1, V_5), \text{mult}(V_5, V_{13}, V_{14}),\\ \text{mult}(V_{15}, V_{16}, V_{17})$$

is a possible choice if choice 1 in ρ_1 is selected and (ii) $K' = \{1, 2, 5\}$, $\theta' = \{V_4/V_1, V_{12}/V_5, V_{13}/V_2\}$ and

$$\overrightarrow{C'} = c(\langle K', \theta'\rangle) = \text{mult}(V_1, V_2, V_3) \leftarrow \text{dec}(V_1, V_5), \text{mult}(V_5, V_2, V_{14})$$

is another possible choice if choice 2 in ρ_1 is selected.

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 (Nienhuys-Cheng and de Wolf 1997; van der Laag 1995) adopted for subsumption order relative to $\bot$.

Lemma 6 *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 18, $\overrightarrow{C}$ and $\overrightarrow{D}$ have the same predicate symbols at the same positions and therefore can be regarded as atoms. It is known (e.g. Theorem 4 in Reynolds 1969) that for any atoms A_1 and A_2 such that $A_1\theta = A_2$, there exists a finite chain of downward covers (involving substitution θ) from A_1 to A_2. Thus, there exists a ρ_1-chain from $\overrightarrow{C}$ to $\overrightarrow{D}$ by repeatedly selecting step 2 in Definition 30. □

Lemma 7 *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, $0 \le j < i$, 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 $\bot$. 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 $\overrightarrow{D} \in \overrightarrow{\mathcal{L}}_\bot$ we can assume that l is mapped to the k-th literal of $\bot$. 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 choice 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 choice 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 6 (and using choice 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 10 *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 6, 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 7, 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 30, 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 31 Let Θ be defined as in Definition 29 and $\theta_1, \theta_2 \in \Theta$. $\theta_1 \subseteq \theta_2$ if and only if there exists a substitution θ such that $\theta_2 = \theta_1\theta$.

Definition 32 Let $\mathcal{K}$ and Θ be defined as in Definition 29 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 $\theta_1 \subseteq \theta_2$. $\langle K_1, \theta_1 \rangle \sim \langle K_2, \theta_2 \rangle$ 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 11 *Let $\mathcal{K}$ and Θ and mapping function c be defined as in Definition 29 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 10 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 32, $\theta_2 = \theta_1\theta$ 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 29). 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 8 *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 ⇒ : 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.

⇐ : 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 22 that $\overrightarrow{C} \succeq_\perp \overrightarrow{D}$ and $\overrightarrow{D} \succeq_\perp \overrightarrow{C}$ and therefore $\overrightarrow{C} \sim_\perp \overrightarrow{D}$. □

Lemma 9 *Let $\mathcal{K}$ and Θ and mapping function c be defined as in Definition 29 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 11 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 8, $c(\langle K \cup \{k\}, \theta\rangle)$ 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 10 *Let $\mathcal{K}$ and Θ, $\top$ and mapping function c be defined as in Definition 29 and $K \in \mathcal{K}$, $\{y/x\}, \theta \in \Theta$ where x and y are distinct variables in the k_1-th and k_2-th literals of $\overrightarrow{\top}$ respectively and k_1-th and k_2-th 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 11 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 8, $c(\langle K, \theta\{y/x\}\rangle)$ 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 12 *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 choice 1 in the definition of ρ_1, then $\overrightarrow{C} \succ_\perp \overrightarrow{D}$ follows from Lemma 9. If it is generated by choice 2 in the definition of ρ_1, then $\overrightarrow{C} \succ_\perp \overrightarrow{D}$ follows from Lemma 10. □

Theorem 13 *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 10 and Theorem 12 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 11, the mapping function c is an order-embedding. The following theorem shows that c is also an order-isomorphism.

Theorem 14 *The mapping function $c : \mathcal{K} \times \Theta \to \overrightarrow{\mathcal{L}}_\perp$ as defined in Definition 29 is an order-isomorphism.*

Proof First we show that the mapping function c is onto. Let $\overrightarrow{C}$ be an ordered clause in $\overrightarrow{\mathcal{L}}_\perp$, then according to Definition 20, there exist substitution θ and selection function s which maps literals of $\overrightarrow{C}\theta$ to equivalent literals from $\overrightarrow{\perp}$. From Definition 29 we have $\overrightarrow{\top}\theta_\top = \overrightarrow{\perp}$ and therefore $\overrightarrow{C}\theta \sqsubseteq \overrightarrow{\top}\theta_\top$ and this implies $\overrightarrow{C} \sqsubseteq \top\theta_\top\theta^{-1}$. 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. Hence, the mapping function c is onto. Moreover, according to Theorem 11, the mapping function c is an order-embedding. Then according to Definition 2, c is an order-isomorphism. □

The proposition below follows directly from Theorem 14.

Proposition 3 *Let $\mathcal{K}$ and Θ and mapping function c be defined as in Definition 29 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 $\langle K, \theta\rangle \sim \langle K', \theta'\rangle$.*

According to Theorem 7 $\langle \overrightarrow{\mathcal{L}}_\perp, \succeq_\perp\rangle$ is a lattice. The proposition below follows directly from Theorem 14 and Remark 4.

Proposition 4 *The mapping function $c : \mathcal{K} \times \Theta \to \overrightarrow{\mathcal{L}}_\perp$ as defined in Definition 29 is a lattice isomorphism and lattices $\langle \overrightarrow{\mathcal{L}}_\perp, \succeq_\perp\rangle$ and $\langle \mathcal{K} \times \Theta, \subseteq\rangle$ are two isomorphic lattices.*

The proposition below follows from c being a lattice isomorphism.

Proposition 5 *Let $\mathcal{K}$ and Θ and mapping function c be defined as in Definition 29 and $K_1, K_2 \in \mathcal{K}$ and $\theta_1, \theta_2 \in \Theta$. Mapping c is join-preserving and meet-preserving that is*:

1. $\mathrm{lgg}_\perp(c(\langle K_1, \theta_1\rangle), c(\langle K_2, \theta_2\rangle)) = c(\langle K_1 \cap K_2, \theta_1 \cap \theta_2\rangle)$,
2. $\mathrm{mgs}_\perp(c(\langle K_1, \theta_1\rangle), c(\langle K_2, \theta_2\rangle)) = c(\langle K_1 \cup K_2, \theta_1 \cup \theta_2\rangle)$.

According to Proposition 5, the least general generalisation ($\mathrm{lgg}_\perp$) and the most general specialisation ($\mathrm{mgs}_\perp$) for $\langle \overrightarrow{\mathcal{L}}_\perp, \succeq_\perp\rangle$ can be defined based on the join 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 $\langle \overrightarrow{\mathcal{L}}_\perp, \mathrm{lgg}_\perp, \mathrm{mgs}_\perp\rangle$ and $\langle \mathcal{K} \times \Theta, \cap, \cup\rangle$ is important from a practical point of view. The construction of the least general generalisation (lgg) of clauses in the general subsumption order is inefficient as the cardinality of the lgg of two clauses can grow very rapidly (see Sect. 6). On the other hand, efficient operators can be implemented for least generalisation and greatest specialisation in the subsumption order relative to a bottom clause. For example, with Plotkin's Relative Least General Generalisation (RLGG), clause length grows exponentially in the number of examples (Plotkin 1971). Hence, an ILP system like Golem (Muggleton and Feng 1990) which uses RLGG is constrained to ij-determinacy to guarantee polynomial-time construction. However, the determinacy restrictions make an ILP system inapplicable in many key application areas, including the learning of chemical properties from atom and bond descriptions. On the other hand, a variant of Plotkin's Relative RLGG which does not need the determinacy restrictions can be designed based on subsumption with respect to a bottom clause. This idea is the basis of a new ILP system which is described in Muggleton et al. (2009).

8 Alternative subsumption orders relative to ⊥

The purpose of the previous sections was to characterise Progol's refinement and the subsumption sub-lattice which is searched by a Progol-like ILP system. We defined sequential subsumption order relative to $\bot$ and studied the properties of this special case of subsumption. In this section we show how other subsumption orders relative to $\bot$ can be defined by using different conditions on the selection functions which define subsequences. For example we show how the first type of incompleteness in Progol's refinement operator can be addressed by relaxing conditions of subsumption order relative to a bottom clause. In this section we define a refinement operator which is less restricted than ρ_1. As demonstrated in Sect. 3, the first type of Progol's refinement incompleteness is due to the choice of ordering of literals in $\bot$ and the fact that clauses are considered as subsequences of $\bot$. This condition was embedded in the definitions of the subsumption order relative to a bottom clause and the refinement operator ρ_1. However, more relaxed conditions can be defined for subsumption and refinement operators relative to $\bot$. 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 $\bot$. Therefore a less restricted ordering can be defined by using a selection function which is injective rather than strictly increasing.

Definition 33 (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 $\overrightarrow{C} \subseteq \overrightarrow{D}$, 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 $\bot$. In the following, we give new definitions for the mapping function and the refinement operator for this less restricted subsumption order.

Definition 34 ($\mathcal{L}_\bot$) Let $\overrightarrow{\bot}$ be the bottom clause as defined in Definition 11. $\overrightarrow{C}$ is in $\mathcal{L}_\bot$ if and only if there exists a substitution θ such that $\overrightarrow{C}\theta$ is an ordered subset of $\overrightarrow{\bot}$.

Definition 35 (Mapping function c') Let $\overrightarrow{\bot}$ and $\mathcal{L}_\bot$ be as defined in Definition 34, n be the number of literals in $\bot$. Let $\overrightarrow{\top}$, $\theta_\top$, θ, Θ, $\mathcal{K}$ be as defined in Definition 29. The mapping function $c' : \mathcal{K} \times \Theta \to \mathcal{L}_\bot$ is defined as follows:

$$c'(\langle K, \theta \rangle) = \left(\bigvee_{i \in K} l_i \text{ where } l_i \text{ is the } i\text{-th literal of } \overrightarrow{\top} \right)\theta.$$

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 36 (ρ_2) Let $\overrightarrow{\bot}$ and $\mathcal{L}_\bot$ be as defined in Definition 34, $\overrightarrow{C}$ be an ordered clause in $\mathcal{L}_\bot$, 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 35. 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 35. $\langle \overrightarrow{C'}, K', \theta' \rangle$ 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_1-th and k_2-th literals of $\overrightarrow{\top}$ respectively and k_1-th and k_2-th are in K'.

The following example demonstrates how the first type of incompleteness (in Example 3) is addressed in ρ_2.

Table 2 Application of ρ_2 in Example 19

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

Example 19 Let $\overrightarrow{C}$ and $\overrightarrow{\bot}$ be as defined in Example 3. 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 2 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:

$$\mathrm{mult}(V_1, V_2, V_3) \leftarrow \mathrm{dec}(V_4, V_5), \mathrm{plus}(V_6, V_7, V_8), \mathrm{plus}(V_9, V_{10}, V_{11}),$$
$$\mathrm{mult}(V_{12}, V_{13}, V_{14}), \mathrm{mult}(V_{15}, V_{16}, V_{17}).$$

This example shows that ρ_2 can address the incompleteness of ρ demonstrated in Example 3. 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 Sect. 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 $\bot$. Therefore, the properties mentioned in Sect. 7, for the mapping function c, also hold for c'. The refinement operator ρ_2 is identical to the refinement operator $\rho_\bot^{(1)}$ introduced in Badea and Stanciu (1999). However, the subsumption order used in Badea and Stanciu (1999) (i.e. weak subsumption) is a special case of the subsumption order introduced in this section.

By different conditions on the selection functions in Definition 33, 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 $\bot$ to be selected more than once. In this case, Definition 33 will be identical to the definition of subsequences considered in Kuwabara et al. (2006). 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. More comparison with the subsumption order introduced in Kuwabara et al. (2006) is given in the next section.

9 Related work and discussion

Progol's refinement operator and its incompleteness with respect to the general subsumption order were initially discussed in Muggleton (1995). The purpose of the present paper was to characterise Progol's refinement space and to give an analysis of the lattice structure and refinement operators for this space. In a previous attempt, Badea and Stanciu (1999) 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. For example, it only characterises the second type of incompleteness but it does not capture the incompleteness due to the ordering of the literals. Note that sequential subsumption implies weak subsumption. This is because if a selection function is strictly increasing then the injectivity property holds which, in turns, entails weak subsumption. As in Badea and Stanciu (1999), we define refinement operators which are based on Laird's operator and defined with respect a bottom clause. However, the approach in Badea and Stanciu (1999) is based on weak subsumption. Moreover, no completeness proof is given in Badea and Stanciu (1999).

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 29. This idea was first used in Tamaddoni-Nezhad and Muggleton (2000) 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.

A subsumption relation for ordered clauses (i.e. ordered subsumption) is studied in Kuwabara et al. (2006). 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 Kuwabara et al. (2006), assumes a mapping function which is monotonically increasing (rather than strictly increasing). As mentioned in the previous section, this leads to a different subsumption order from the one considered in this paper (i.e. sequential subsumption) and the results from this paper are not applicable. In the context of data mining from sequential data, SeqLog (Lee and De Raedt 2003) is defined as a logical language for representing and reasoning about sequential data. Subsumption relations are defined in Lee and De Raedt (2003) for simple and complex sequences and an optimal refinement operator is given for SeqLog. The frameworks used in Kuwabara et al. (2006) and Lee and De Raedt (2003) can be viewed as general cases for ordered and sequential subsumption. Whereas in this paper we introduce subsumption order relative to a bottom clause. The refinement operators in this paper are also different as they are defined with respect to a bottom clause. We proved the existence of operators $\mathrm{lgg}_{\perp}$ and $\mathrm{mgs}_{\perp}$ with respect to a bottom clause. It was shown that the subsumption order relative to a bottom clause defines a lattice and this lattice is isomorphic to an atomic lattice. Moreover, unlike the general subsumption and the sequential subsumption which are NP-complete, subsumption relative to a bottom clause can be decided in polynomial time.

The theoretical results presented in this paper are applicable to ILP systems such as Progol and Aleph which use some form of Inverse Entailment (IE). Moreover, these results are also applicable to other ILP systems which use a bottom clause to restrict the search space. These include ILP systems which use stochastic algorithms to explore the hypothesis space bounded by a bottom clause (e.g. Srinivasan 2000; Tamaddoni-Nezhad and Muggleton 2000; Zelezny et al. 2003; Muggleton and Tamaddoni-Nezhad 2007; Duboc et al. 2008). The search space of these systems can be characterised by $\overrightarrow{\mathcal{L}}_{\perp}$ (or a subset of $\overrightarrow{\mathcal{L}}_{\perp}$). Note that a refinement operator cannot be both complete and non-redundant. For ILP systems such as Progol and Aleph (which use a top-down graph based search) we only need the refinement operators to be weakly complete and preferably non-redundant. However, for the stochastic algorithms (where the search could start from any point in the search space) we need complete refinement operators even if this leads to redundancy. The refinement operator ρ_0 which was described in this paper is weakly complete and non-redundant. This operator has been implemented in Progol. The refinement operator ρ_1 is complete but redundant and it has been the basis for the stochastic refinement operators in Progol (Tamaddoni-Nezhad and Muggleton 2002; Muggleton and Tamaddoni-Nezhad 2007). Moreover, in this paper we have shown that, unlike for the general subsumption order, efficient lgg operators can be designed for $\overrightarrow{\mathcal{L}}_{\perp}$. This idea is the basis of a new ILP system which implements efficient asymmetric relative minimal generalisations for the subsumption order relative to a bottom clause (Muggleton et al. 2009).

10 Conclusions

In this paper we have studied the lattice structure and refinement operators for the hypothesis space bounded by a most specific (bottom) clause. We introduced a subsumption order relative to a bottom clause and demonstrated how clause refinement in a Progol-like ILP system can be characterised with respect to this order. We proved that least general generalisation (lgg) and most general specialisation (mgs) exist for subsumption order relative to a bottom clause and that this order defines a lattice which is isomorphic to an atomic lattice. We also proved that ideal refinement operators exist for this order. The theoretical results presented in this paper are applicable to ILP systems which use Inverse Entailment (IE) as well as other systems which use a bottom clause to restrict the search space. This is important for better understanding of the constrained refinement space of these systems. Moreover, characterising this refinement sub-lattice can lead to more efficient ILP algorithms and operators for searching this particular sub-lattice. For example, it is shown that, unlike for the general subsumption order, efficient least generalisation operators can be designed for the subsumption order relative to a bottom clause.

Acknowledgements We would like to thank the four 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.

Appendix A: Progol's definite mode language and algorithm for constructing the bottom clause

The following definitions describe Progol's mode declaration (M), definite mode language ($\mathcal{L}(M)$), depth-bounded mode language ($\mathcal{L}_i(M)$) and Progol's algorithm for constructing the bottom clause ($\perp_i$).

Definition 37 (Mode declaration M) A mode declaration has either the form modeh(n,atom) or modeb(n,atom) where n, the recall, is either an integer, $n > 1$, or '*' and atom is a ground atom. Terms in the atom are either normal or place-marker. A normal term is either a constant or a function symbol followed by a bracketed tuple of terms. A place-marker is either +type, −type or #type, where type is a constant. If m is a mode declaration then $a(m)$ denotes the atom of m with place-markers replaced by distinct variables. The sign of m is positive if m is a modeh and negative if m is a modeb.

Definition 38 (Definite mode language $\mathcal{L}(M)$) Let C be a definite clause with a defined total ordering over the literals and M be a set of mode declarations. $C = h \leftarrow b_1, \dots, b_n$ is in the definite mode language $\mathcal{L}(M)$ if and only if (1) h is the atom of a modeh declaration in M with every place-marker +type and −type replaced by variables and every place-marker

#type replaced by a ground term and (2) every atom b_i in the body of C is the atom of a modeb declaration in M with every place-marker +type and −type replaced by variables and every place-marker #type replaced by a ground term and (3) every variable of +type in any atom b_i is either of +type in h or of −type in some atom b_j, $1 \leq j < i$.

Definition 39 (Depth of variables) Let C be a definite clause and v be a variable in C. Depth of v is defined as follows:

$$d(v) = \begin{cases} 0 & \text{if } v \text{ is in the head of } C \\ (\max_{u \in U_v} d(u)) + 1 & \text{otherwise} \end{cases}$$

where U_v are the variables in atoms in the body of C containing v.

Definition 40 (Depth-bounded mode language $\mathcal{L}_i(M)$) Let C be a definite clause with a defined total ordering over the literals and M be a set of mode declarations. C is in $\mathcal{L}_i(M)$ if and only if C is in $\mathcal{L}(M)$ and all variables in C have depth at most i according to Definition 39.

Algorithm 2 (Algorithm for constructing $\perp_i$)

1. Given natural numbers h, i, Horn clauses B, definite clause e and set of mode declarations M.
2. Let $k = 0$, hash : Terms $\to N$ be a hash function which uniquely maps terms to natural numbers, $\overline{e}$ be the clause normal form logic program $\overline{a} \wedge b_1 \wedge \cdots \wedge b_n$, $\perp_i = \langle\rangle$ and InTerms$= \emptyset$.
3. If there is no modeh in M such that $a(m) \preceq a$ then return $\Box$. Otherwise let m be the first modeh declaration in M such that $a(m) \preceq a$ with substitution θ_h. Let a_h be a copy of $a(m)$ and for each v/t in θ_h if v corresponds to a #type in m then replace v in a_h by t otherwise replace v in a_h by v_k where $k = \text{hash}(t)$ and add v to InTerms if v corresponds to +type. Add a_h to $\perp_i$.
4. If $k = i$ return $\perp_i$ else $k = k + 1$.
5. For each modeb m in M let $\{v_1, \ldots, v_n\}$ be the variables of +type in $a(m)$ and $T(m) = T_1 \times \cdots \times T_n$ be a set of n-tuples of terms such that each T_i corresponds to the set of all terms of the type associated with v_i in m (term t is tested to be of a particular type by calling Prolog with type(t) as goal). For each $\langle t_1, \ldots, t_n\rangle$ in $T(m)$ let a_b be a copy of $a(m)$ and $\theta = \{v_1/t_1, \ldots, v_n/t_n\}$. If Prolog with depth-bound h succeeds on goal $a_b\theta$ with the set of answer substitutions Θ_b then for each θ_b in Θ_b and for each v/t in θ_b if v corresponds to a #type in m then replace v in a_b by t otherwise replace v in a_b by v_k where $k = \text{hash}(t)$ and add v to InTerms if v corresponds to −type. Add $\overline{a_b}$ to $\perp_i$.
6. Goto step 4.

Appendix B: Morphism between the function free atomic lattice and the lattice of variable partitions

It was shown in Sect. 6 that the subsumption order relative to a bottom clause defines a lattice and this lattice is isomorphic to an atomic lattice. The complexity of the hypothesis space bounded by a bottom clause can be further analysed by mapping the lattice $\langle \overrightarrow{\mathcal{L}}_\perp, \succeq_\perp\rangle$ to a partition lattice. In the following, we show the morphism between the function free atomic lattice and the lattice of variable partitions.

Definition 41 Let Π_n be the set of all partitions on $\{1, 2, \dots, n\}$ and P_1 and P_2 be partitions in Π_n. We say P_1 is finer than P_2, denoted by $P_1 \leq P_2$ if and only if for each block B_1 in P_1 there is a block B_2 in P_2 such that $B_1 \subseteq B_2$. P_1 is properly finer than P_2, denoted by $P_1 < P_2$, if $P_1 \leq P_2$ and $P_2 \not\leq P_1$.

It is known (Davey and Priestley 2002) that Π_n is partially ordered by $\leq$ and that $\langle \Pi_n, \leq \rangle$ is a lattice.

Proposition 6 *Let Π_n and $\leq$ be as defined in Definition 41. Then $\langle \Pi_n, \leq \rangle$ is a lattice.*

Definition 42 (Mapping function P) Let $\mathcal{A}_n$ be the set of all atoms in a language with only one n-ary predicate symbol and with no constant and function symbol. The mapping function $P : \mathcal{A}_n \to \Pi_n$ is defined to map any atom $A = p(v_1, v_2, \dots, v_n)$ in $\mathcal{A}_n$ to a partition π in Π_n such that for each block B in π, $\{i, j\} \subseteq B$ if and only if variables v_i and v_j are the same.

Example 20 Let $A = p(X, Y, X, Z, Y, X)$ be an atom in $\mathcal{A}_6$. Then $P(A) = \{\{1, 3, 6\}, \{2, 5\}, \{4\}\}$.

Lemma 11 *Let $\mathcal{A}_n$ be as defined in Definition 42 and $A_1 = p(u_1, \dots, u_n)$ and $A_2 = p(v_1, \dots, v_n)$ be atoms in $\mathcal{A}_n$. There exists a variable substitution θ such that $A_1\theta = A_2$ if and only if for any pair of variables u_i and u_j in A_1 if u_i and u_j are the same then variables v_i and v_j in A_2 are the same.*

Proof $\Rightarrow$: Suppose that there exists a variable substitution θ such that $p(u_1, \dots, u_n)\theta = p(v_1, \dots, v_n)$. Let $\{u_i/v_i, u_j/v_j\} \subseteq \theta$. Then according to Definition 7, u_i and u_j must be distinct variables. Hence, if variables u_i and u_j are the same then variables v_i and v_j are the same.

$\Leftarrow$: Suppose that for any pair of variables u_i and u_j in A_1 if u_i and u_j are the same then variables v_i and v_j in A_2 are the same. Then a mapping function can be defined which maps each variable u_i from A_1 to a variable v_i from A_2. Then according to Definition 7, there is a variable substitution θ such that $A_1\theta = A_2$. □

Example 21 Let A_1 and A_2 be atoms in $\mathcal{A}_6$ as defined below

$$A_1 = p(X_1, Y_1, X_1, Z_1, Y_1, X_1),$$
$$A_2 = p(X_2, Y_2, X_2, X_2, Y_2, X_2).$$

We have $A_1\theta = A_2$ where $\theta = \{X_1/X_2, Y_1/Y_2, Z_1/X_2\}$. For any pair of variables u_i and u_j in A_1 if u_i and u_j are the same then variables v_i and v_j in A_2 are the same. For example, u_1 and u_3 in A_1 represent the same variable X_1 and v_1 and v_3 in A_2 represent the same variable X_2.

Theorem 15 *Let $\mathcal{A}_n$ be as defined in Definition 42 and A_1 and A_2 be atoms in $\mathcal{A}_n$. $A_1 \succeq A_2$ if and only if $P(A_1) \leq P(A_2)$.*

Proof $\Rightarrow$: Let $A_1 = p(u_1, \dots, u_n)$ and $A_2 = p(v_1, \dots, v_n)$ such that $A_1 \succeq A_2$. Then according to Definition 9, there exists a substitution θ such that $p(u_1, \dots, u_n)\theta = p(v_1, \dots, v_n)$.

According to Lemma 11, for any pair of variables u_i and u_j representing the same variable in A_1, variables v_i and v_j represent the same variable in A_2. Then according to Definition 42, if $\{i, j\} \subseteq B_1$ where $B_1 \in P(A_1)$ then there is $\{i, j\} \subseteq B_2$ where $B_2 \in P(A_2)$. Then according to Definition 41, $P(A_1) \leq P(A_2)$.

$\Leftarrow$: Let $A_1 = p(u_1, \dots, u_n)$ and $A_2 = p(v_1, \dots, v_n)$ such that $P(A_1) \leq P(A_2)$. Then according to Definition 41 for each block B_1 in $P(A_1)$ there is a block B_2 in $P(A_2)$ such that $B_1 \subseteq B_2$. Hence, for each $\{i, j\} \subseteq B_1$ where $B_1 \in P(A_1)$, there is $\{i, j\} \subseteq B_2$ where $B_2 \in P(A_2)$. Then according to Definition 42, for any pair of variables u_i and u_j representing the same variable in A_1, variables v_i and v_j represent the same variable in A_2. According to Lemma 11, there exists a substitution θ such that $p(u_1, \dots, u_n)\theta = p(v_1, \dots, v_n)$ and therefore $A_1 \succeq A_2$. □

Theorem 16 *The mapping function $P : \mathcal{A}_n \to \Pi_n$ as defined in Definition 42 is an order-isomorphism.*

Proof First we show that the mapping function P is onto. Let π be a partition in Π_n. We show that there is an atom A in $\mathcal{A}_n$ such that $P(A) = \pi$. Let $A = p(v_1, v_2, \dots, v_n)$ be an atom in $\mathcal{A}_n$ such that for each block B in π and for each $\{i, j\} \subseteq B$, variables v_i and v_j are the same. Then according to Definition 42 we have $P(A) = \pi$ and therefore P is onto. Moreover, according to Theorem 15 and Definition 2, the mapping function P is order-embedding. Then according to Definition 2, P is an order-isomorphism. □

According to Proposition 6, $\langle \Pi_n, \leq \rangle$ is a lattice. The proposition below follows directly from Theorem 16 and Remark 4.

Proposition 7 *The mapping function $P : \mathcal{A}_n \to \Pi_n$ as defined in Definition 42 is a lattice isomorphism and lattices $\langle \Pi_n, \leq \rangle$ and $\langle \mathcal{A}_n, \succeq \rangle$ are two isomorphic lattices.*

References

Badea, L., & Stanciu, M. (1999). Refinement operators can be (weakly) perfect. In *Proceedings of the 9th international workshop on inductive logic programming* (Vol. 1634, pp. 21–32).

Davey, B.A., & Priestley, H. A. (2002). *Introduction to lattices and order*. Cambridge: Cambridge University Press.

Duboc, A. L., Paes, A., & Zaverucha, G. (2008). Using the bottom clause and mode declarations on FOL theory revision from examples. In *Proceedings of the 18th international conference on inductive logic programming* (pp. 91–106). Berlin: Springer.

Garey, M. R., & Johnson, D. S. (1979). *Computers and intractability: a guide to the theory of NP-completeness*. New York: Freeman.

Kuwabara, M., Ogawa, T., Hirata, K., & Harao, M. (2006). On generalization and subsumption for ordered clauses. In *Proceedings of the JSAI 2005 Workshops* (pp. 212–223).

Laird, P. D. (1987). *Learning from good data and bad*. PhD thesis, Yale University.

Lee, S. D., & De Raedt, L. (2003). Constraint based mining of first order sequences in SeqLog. In *Database Support for Data Mining Applications* (pp. 155–176).

Muggleton, S. (1995). Inverse entailment and Progol. *New Generation Computing*, *13*, 245–286.

Muggleton, S., Santos, J., & Tamaddoni-Nezhad, A. (2009). Towards an ILP system based on asymmetric relative minimal generalisation. In *Proceedings of the 19th international conference on inductive logic programming* (to appear).

Muggleton, S. H., & Feng, C. (1990). Efficient induction of logic programs. In *Proceedings of the first conference on algorithmic learning theory, Tokyo, Ohmsha* (pp. 368–381).

Muggleton, S. H., & Tamaddoni-Nezhad, A. (2007). QG/GA: A stochastic search for Progol. *Machine Learning*, *70*(2–3), 123–133.

Nienhuys-Cheng, S.-H., & de Wolf, R. (1997). *LNAI*: Vol. *1228*. *Foundations of inductive logic programming*. Berlin: Springer.

Plotkin, G. D. (1971). *Automatic methods of inductive inference*. PhD thesis, Edinburgh University.

Reynolds, J. C. (1969). Transformational systems and the algebraic structure of atomic formulas. In B. Meltzer & D. Michie (Eds.), *Machine intelligence 5* (pp. 135–151). Edinburgh: Edinburgh University Press.

Rouveirol, C. (1992). Extensions of inversion of resolution applied to theory completion. In S. Muggleton (Ed.), *Inductive logic programming*. London: Academic Press.

Srinivasan, A. (2000). *A study of two probabilistic methods for searching large spaces with ilp*. Technical Report PRG-TR-16-00, Oxford University Computing Laboratory, Oxford.

Srinivasan, A. (2007). *The Aleph manual*. Oxford: University of Oxford.

Tamaddoni-Nezhad, A., & Muggleton, S. H. (2000). Searching the subsumption lattice by a genetic algorithm. In J. Cussens & A. Frisch (Eds.), *Proceedings of the 10th international conference on inductive logic programming* (pp. 243–252). Berlin: Springer.

Tamaddoni-Nezhad, A., & Muggleton, S. H. (2002). A genetic algorithms approach to ILP. In *Proceedings of the 12th international conference on inductive logic programming* (pp. 285–300). Berlin: Springer.

Tamaddoni-Nezhad, A., & Muggleton, S. H. (2008). A note on refinement operators for IE-based ILP systems. In *LNAI: Vol. 5194*. *Proceedings of the 18th international conference on inductive logic programming* (pp. 297–314). Berlin: Springer.

van der Laag, P. (1995). *An analysis of refinement operators in inductive logic programming*. Tinbergen institute research series, Rotterdam.

van der Laag, P. R. J., & Nienhuys-Cheng, S. H. (1994). Existence and nonexistence of complete refinement operators. In *Machine learning. ECML-94: European conference on machine learning, Catania, Italy, April 6–8, 1994: Proceedings* (pp. 307–322). Berlin: Springer.

Zelezny, F., Srinivasan, A., & Page, D. (2003). Lattice-search runtime distributions may be heavy-tailed. In S. Matwin & C. Sammut (Eds), *Proceedings of the 12th international conference on inductive logic programming* (Vol. 2583, pp. 333–345).