Generality Relations in Answer Set Programming

Katsumi Inoue[1] and Chiaki Sakama[2]

[1] National Institute of Informatics
2-1-2 Hitotsubashi, Chiyoda-ku, Tokyo 101-8430, Japan
ki@nii.ac.jp
[2] Department of Computer and Communication Sciences, Wakayama University
Sakaedani, Wakayama 640-8510, Japan
sakama@sys.wakayama-u.ac.jp

Abstract. This paper studies generality relations on logic programs. Intuitively, a program P_1 is *more general* than another program P_2 if P_1 gives us more information than P_2. In this paper, we define various kinds of generality relations over nonmonotonic programs in the context of *answer set programming*. The semantic properties of generality relations are investigated based on domain theory, and both a minimal upper bound and a maximal lower bound are constructed for any pair of logic programs. We also introduce the concept of *strong generality* between logic programs and investigate its relationships to strong equivalence. These results provide a basic theory to compare the degree of incompleteness between nonmonotonic logic programs, and also have important applications to inductive logic programming and multi-agent systems.

1 Introduction

Nonmonotonic logic programs, or logic programs with negation as failure and/or disjunctions, are useful for representing incomplete knowledge and partial information. To judge whether two logic programs represent the same knowledge, the notion of *equivalence* has recently become important in logic programming [7,6]. Another useful measure to compare the amount of information brought by logic programs is the concept of *generality*. Intuitively, a logic program P_1 is considered *more general* than another logic program P_2 if P_1 gives us more information than P_2.

The generality notion is important in the field of *inductive logic programming*, and basic studies have been done in this context [10,8,9] for *monotonic* logic programs, which can be defined as subsets of first-order clausal theories. Model theoretically, given two monotonic programs P_1 and P_2, the situation that P_1 is more general than P_2 is represented as $P_1 \models P_2$, that is, P_1 entails P_2, which means that every model of P_1 is also a model of P_2. For instance, the program $\{ p \leftarrow \}$ is more general than the program $\{ p \leftarrow q \}$.

In the context of *nonmonotonic* logic programs, however, relatively little attention is given to generality relations although the equivalence notion has been

S. Etalle and M. Truszczyński (Eds.): ICLP 2006, LNCS 4079, pp. 211–225, 2006.

studied in depth. Let us intuitively define that, for two nonmonotonic programs P_1 and P_2, P_1 is more general than P_2 if P_1 entails more information than P_2 under the canonical model semantics (e.g., *answer set semantics* [3]). Unfortunately, there is a difficulty in this definition such that a nonmonotonic program generally has multiple canonical models. This is contrasted to a monotonic program that has a unique canonical model or an "extension" as the logical consequences of the program. For instance, consider two nonmonotonic programs:

$$\begin{aligned} P_1 :\ & p \leftarrow not\, q\,, \\ P_2 :\ & p \leftarrow not\, q\,, \\ & q \leftarrow not\, p\,. \end{aligned}$$

Here, P_1 has the single answer set $\{p\}$ and P_2 has two answer sets $\{p\}$ and $\{q\}$. If we reason *skeptically* and draw conclusions from the intersection of all answer sets, P_1 entails p but P_2 entails nothing. As a result, P_1 is considered more informative and more general than P_2. By contrast, if we reason *credulously* and draw conclusions from the union of all answer sets, P_2 is considered more informative than P_1. Thus, the result depends on the type of inference.

In this paper, we study a theory to compare the degree of incomplete information brought by nonmonotonic logic programs in the framework of *answer set programming*. By the above discussion, it is more appropriate to focus on the whole collection of answer sets of a program than on the set of literals entailed from it. Then, to compare the infomation contents of two logic programs, it is natural to directly compare the collections of answer sets of the two programs. For this purpose, *domain theory* [11,16,4], which studies orderings over the powerset of a domain, is particularly convenient. There are at least two reasonable philosophies to judge that one description is more informative than another description in domain theory. Suppose, for example, that there are three descriptions about the contents of a bag, which are represented by the following logic programs:

$$\begin{aligned} P_1 :\ & \mathit{red\_fruit}\,;\, \mathit{yellow\_fruit} \leftarrow\,, \\ P_2 :\ & \mathit{cherry}\,;\, \mathit{strawberry} \leftarrow\,, \\ & \mathit{red\_fruit} \leftarrow \mathit{cherry}\,, \\ & \mathit{red\_fruit} \leftarrow \mathit{strawberry}\,, \\ P_3 :\ & \mathit{cherry}\,;\, \mathit{banana}\,;\, \mathit{purple\_fruit} \leftarrow\,, \\ & \mathit{red\_fruit} \leftarrow \mathit{cherry}\,, \\ & \mathit{yellow\_fruit} \leftarrow \mathit{banana}\,. \end{aligned}$$

Then, P_2 is more informative than P_1 in the sense that both *cherry* and *strawberry* provide further restrictions on the contents by ruling out the possibility of *yellow_fruit* as well as other *red_fruit* like *apple*, for example. We will represent this situation as $P_2 \models^\sharp P_1$ meaning that, for each answer set S of P_2, there is an answer set T of P_1 such that $T \subseteq S$. The relation $\models^\sharp$ is called the *Smyth ordering*. On the other hand, P_3 is more informative than P_1 in the sense that P_3 provides a further enumeration of positive assertions which does not rule out the

possibility of *purple_fruit* like *grape*, for example. We will represent this situation as $P_3 \models^\flat P_1$ meaning that, for each answer set T of P_1, there is an answer set S of P_3 such that $T \subseteq S$. The relation $\models^\flat$ is called the *Hoare ordering*. Then, both a minimal upper bound and a maximal lower bound are constructed for any pair of logic programs with respect to each generality ordering. We will also relate these two generality orderings with the generality relations with respect to skeptical and credulous entailment, respectively. Furthermore, we will introduce the concept of *strong generality* between logic programs and investigate its relationship to *strong equivalence* [6].

The rest of this paper is organized as follows. After introducing basic concepts of answer set programming and domain theory, Section 2 presents a theory of generality in logic programs. Section 3 examines minimal upper and maximal lower bounds of logic programs with respect to generality orderings, and discusses how to compute logic programs whose answer sets exactly correspond to those bounds. Section 4 relates the generality relations with skeptical and credulous entailment in answer set programming. Section 5 defines the notion of strong generality and relates it with strong equivalence of logic programs. Section 6 discusses applications of generality relations to *inductive logic programming* and *multi-agent systems* as well as related work.

2 Generality Relations over Answer Sets

2.1 Extended Disjunctive Programs

A *program* considered in this paper is an *extended disjunctive program* (EDP) which is a set of *rules* of the form:

$$L_1 ; \cdots ; L_l \leftarrow L_{l+1}, \ldots, L_m, \mathit{not}\, L_{m+1}, \ldots, \mathit{not}\, L_n \quad (n \geq m \geq l \geq 0) \qquad (1)$$

where each L_i is a literal, *not* is *negation as failure* (NAF), and ";" represents disjunction. The left-hand side of a rule is the *head*, and the right-hand side is the *body*. A rule is *disjunctive* if its head contains more than one literal. A rule is an *integrity constraint* if its head is empty, and is a *fact* if its body is empty. An EDP is called an *extended logic program* (ELP) if $l \leq 1$ for each rule (1). A program is *NAF-free* if every rule contains no *not*, i.e., $m = n$ for each rule (1). A program with variables is semantically identified with its ground instantiation.

In this paper, we consider the *answer set semantics* for EDPs [3]. Let *Lit* be the set of all ground literals in the language of programs. A set $S\,(\subseteq \mathit{Lit})$ *satisfies* a ground rule of the form (1) if $\{L_{l+1}, \ldots, L_m\} \subseteq S$ and $\{L_{m+1}, \ldots, L_n\} \cap S = \emptyset$ imply $L_i \in S$ for some i $(1 \leq i \leq l)$. Let P be an NAF-free EDP. Then, a set $S(\subseteq \mathit{Lit})$ is an *answer set* of P if S is a minimal set such that

1. S satisfies every rule from the ground instantiation of P,
2. $S = \mathit{Lit}$ if S contains a pair of *complementary literals*, L and $\neg L$.

Next, let P be any EDP and $S \subseteq \mathit{Lit}$. For every rule of the form (1) in the ground instantiation of P, the rule $L_1; \cdots; L_l \leftarrow L_{l+1}, \ldots, L_m$ is included in the

NAF-free program P^S iff $\{L_{m+1}, \ldots, L_n\} \cap S = \emptyset$. Then, S is an *answer set* of P if S is an answer set of P^S. The set of all answer sets of P is written as $A(P)$. An answer set is *consistent* if it is not Lit. A program P is *consistent* if it has a consistent answer set; otherwise, P is *inconsistent*. An inconsistent program is called *contradictory* if it has the single answer set Lit, and is called *incoherent* if it has no answer set.

We will see that the following two notions of equivalence are important to develop a theory of generality in answer set programming.

Definition 2.1. Let P and Q be programs. P and Q are *weakly equivalent* if $A(P) = A(Q)$ holds. On the other hand, P and Q are *strongly equivalent* [6] if for any logic program R, $A(P \cup R) = A(Q \cup R)$ holds.

For example, $P = \{\, p \leftarrow not\, q, \quad q \leftarrow not\, p \,\}$ and $Q = \{\, p; q \leftarrow \,\}$ are weakly equivalent, but not strongly equivalent.

2.2 Ordering on Powersets

We first recall some mathematical definitions about domains [4]. A *pre-order* $\sqsubseteq$ is a binary relation which is reflexive and transitive. A pre-order $\sqsubseteq$ is a *partial order* if it is also anti-symmetric. A *pre-ordered set* (resp. *partially ordered set*; *poset*) is a set D with a pre-order (resp. partial order) $\sqsubseteq$ on D.

For any pre-ordered set $\langle D, \sqsubseteq \rangle$, a poset is induced over the equivalence classes of D. That is, for any element $X \in D$, define the equivalence class as

$$[X] = \{\, Y \in D \mid Y \sqsubseteq X,\ X \sqsubseteq Y \,\}.$$

The equivalence relation partitions D into a set of disjoint equivalence classes. Introducing the relation $\preceq$ on the set of these equivalence classes as:

$$[X] \preceq [Y] \quad \text{if} \quad X \sqsubseteq Y,$$

the relation $\preceq$ becomes a partial order on the set.

For any set D, let $\mathcal{P}(D)$ be the powerset of D. Given a poset $\langle D, \sqsubseteq \rangle$ and $X, Y \in \mathcal{P}(D)$, the *Smyth order* is defined as

$$X \models^\sharp Y \quad \text{iff} \quad \forall x \in X\, \exists y \in Y.\, y \sqsubseteq x\,,$$

and the *Hoare order* is defined as

$$X \models^\flat Y \quad \text{iff} \quad \forall y \in Y\, \exists x \in X.\, y \sqsubseteq x\,.$$

The relations $\models^\sharp$ and $\models^\flat$ are pre-orders on $\mathcal{P}(D)$. Note that the orderings $\models^\sharp$ and $\models^\flat$ are slightly different from the standard ones: we allow the empty set $\emptyset$ ($\in \mathcal{P}(D)$) as the top element $\top^\sharp$ in $\langle \mathcal{P}(D), \models^\sharp \rangle$ and the bottom element $\bot^\flat$ in $\langle \mathcal{P}(D), \models^\flat \rangle$. This is because we will associate $\emptyset$ with the class of incoherent programs so that we enable comparison of all classes of EDPs.

Example 2.1. Consider the poset $\langle \mathcal{P}(\{p, q\}), \subseteq \rangle$. Then, we have $\{\{p, q\}\} \models^\sharp \{\{p\}\}$ and $\{\{p\}\} \models^\sharp \{\{p\}, \{q\}\}$, and hence $\{\{p, q\}\} \models^\sharp \{\{p\}, \{q\}\}$. On the other hand, $\{\{p, q\}\} \models^\flat \{\{p\}, \{q\}\}$ but $\{\{p\}, \{q\}\} \models^\flat \{\{p\}\}$. Note that both $\{\emptyset, \{p\}\} \models^\sharp \{\emptyset, \{q\}\}$ and $\{\emptyset, \{q\}\} \models^\sharp \{\emptyset, \{p\}\}$ hold, indicating that $\models^\sharp$ is not a partial order.

In the following, we assume a poset $\langle D, \sqsubseteq \rangle$ such that the domain $D = \mathcal{P}(Lit)$ is the family of subsets of Lit, i.e., the class of sets of literals in the language and the partial-order $\sqsubseteq$ is the subset relation $\subseteq$. Then, the Smyth and Hoare orderings are defined on $\mathcal{P}(\mathcal{P}(Lit))$, which enables us to order sets of literals or sets of answer sets. In particular, both $\langle \mathcal{P}(\mathcal{P}(Lit)), \models^\sharp \rangle$ and $\langle \mathcal{P}(\mathcal{P}(Lit)), \models^\flat \rangle$ are pre-ordered sets. Moreover, if we associate an EDP P with its set of answer sets $A(P)$, the ordering on the EDPs becomes possible as follows.

Definition 2.2. Given the poset $\langle \mathcal{P}(Lit), \subseteq \rangle$ and two programs P, Q that are constructed in the same language with Lit, we define:

$$P \models^\sharp Q \quad \text{if} \quad A(P) \models^\sharp A(Q),$$
$$P \models^\flat Q \quad \text{if} \quad A(P) \models^\flat A(Q).$$

We say that P is *more* $\sharp$*-general* (resp. *more* $\flat$*-general*) than Q if $P \models^\sharp Q$ (resp. $P \models^\flat Q$).

Intuitively, $\sharp$-generality and $\flat$-generality reflect the following situations. $P \models^\sharp Q$ means that any answer set of P is more (or equally) informative than some answer set of Q. On the other hand, $P \models^\flat Q$ means that any answer set of Q is less (or equally) informative than some answer set of P. When both P and Q have single answer sets, it is obvious that $P \models^\sharp Q$ iff $P \models^\flat Q$. The notion of $\sharp$-generality has been introduced in [5] such that Q is defined to be *weaker* than P if $P \models^\sharp Q$, although properties of this ordering have never been deeply investigated so far.

Both $\sharp$-generality and $\flat$-generality are naturally connected to the notion of weak equivalence in answer set programming.

Theorem 2.1. *Let P and Q be EDPs. Then, the following three are equivalent:*

(1) $P \models^\sharp Q$ *and* $Q \models^\sharp P$;
(2) $P \models^\flat Q$ *and* $Q \models^\flat P$;
(3) P *and* Q *are weakly equivalent.*

Proof. We prove (1)$\Leftrightarrow$(3) but (2)$\Leftrightarrow$(3) can be proved in the same way.

$P \models^\sharp Q$ and $Q \models^\sharp P$
iff $\forall S \in A(P)\, \exists T \in A(Q).\, T \subseteq S$ and $\forall T \in A(Q)\, \exists S \in A(P).\, S \subseteq T$
iff $\forall S \in A(P)\, \exists T \in A(Q)\, \exists S' \in A(P).\, S' \subseteq T \subseteq S$ and $\forall T \in A(Q)\, \exists S \in A(P)\, \exists T' \in A(Q).\, T' \subseteq S \subseteq T$
iff $\forall S \in A(P)\, \exists T \in A(Q).\, T = S$ and $\forall T \in A(Q)\, \exists S \in A(P).\, S = T$ (because for any two answer sets $S, T \in A(P)$, $S \subseteq T$ implies $S = T$ by the fact that $A(P)$ is an anti-chain on the poset $\langle \mathcal{P}(Lit), \subseteq \rangle$)
iff $\forall S \in A(P).\, S \in A(Q)$ and $\forall T \in A(Q).\, T \in A(P)$
iff $A(P) \subseteq A(Q)$ and $A(Q) \subseteq A(P)$ iff $A(P) = A(Q)$. □

By Theorem 2.1, for any EDP P, every EDP in the equivalence class $[P]$ induced by the pre-order $\models^\sharp$ or $\models^\flat$ is weakly equivalent to P.

Example 2.2. Consider the following programs:

$$\begin{aligned} P_1 &: p \leftarrow not\, q, \\ P_2 &: p \leftarrow not\, q, \\ &\quad q \leftarrow not\, p, \\ P_3 &: p\,;\, q \leftarrow , \\ P_4 &: p\,;\, q \leftarrow , \\ &\quad p \leftarrow q, \\ &\quad q \leftarrow p. \end{aligned}$$

Then, $P_4 \models^\sharp P_1 \models^\sharp P_2$, and $P_4 \models^\flat P_2 \models^\flat P_1$ (see Example 2.1). P_2 and P_3 are weakly equivalent, and thus $P_2 \models^\sharp P_3 \models^\sharp P_2$ and $P_2 \models^\flat P_3 \models^\flat P_2$.

3 Minimal Upper and Maximal Lower Bounds

In this section, we show that both a minimal upper bound and a maximal lower bound of any pair of logic programs exist with respect to generality orderings, and discuss how to compute logic programs whose answer sets exactly correspond to those bounds. Those bounds are important in the theory of generalization and specialization in inductive logic programming [10]. In the following, let $\mathcal{EDP}$ be the class of all EDPs which can be constructed in the language.

Proposition 3.1. *Both* $\langle \mathcal{EDP}, \models^\sharp \rangle$ *and* $\langle \mathcal{EDP}, \models^\flat \rangle$ *are pre-ordered sets.*

For notational convenience, we denote the $\sharp$- or $\flat$-generality relation as $\models^{\sharp/\flat}$ when distinction between $\sharp$- and $\flat$-general orderings is not important. In what follows, we consider the problem to find a *minimal upper bound* (mub) and a *maximal lower bound* (mlb) of given two programs P_1 and P_2. Because $\langle \mathcal{EDP}, \models^{\sharp/\flat} \rangle$ is only a pre-ordered set, there is no unique minimal/maximal bound in general. In Section 3.2, however, it is shown that the *least upper bound* (lub) and the *greatest lower bound* (glb) can be constructed for the equivalence classes $[P_1]$ and $[P_2]$ under these orderings.

3.1 Mub and Mlb in Smyth and Hoare Orderings

In this section, we suppose that $P_1, P_2 \in \mathcal{EDP}$.

Definition 3.1. A program $Q \in \mathcal{EDP}$ is an *upper bound* of P_1 and P_2 in $\langle \mathcal{EDP}, \models^{\sharp/\flat} \rangle$ if $Q \models^{\sharp/\flat} P_1$ and $Q \models^{\sharp/\flat} P_2$. An upper bound Q is an *mub* of P_1 and P_2 in $\langle \mathcal{EDP}, \models^{\sharp/\flat} \rangle$ if for any upper bound Q', $Q \models^{\sharp/\flat} Q'$ implies $Q' \models^{\sharp/\flat} Q$.

On the other hand, $Q \in \mathcal{EDP}$ is a *lower bound* of P_1 and P_2 in $\langle \mathcal{EDP}, \models^{\sharp/\flat} \rangle$ if $P_1 \models^{\sharp/\flat} Q$ and $P_2 \models^{\sharp/\flat} Q$. A lower bound Q is an *mlb* of P_1 and P_2 in $\langle \mathcal{EDP}, \models^{\sharp/\flat} \rangle$ if for any lower bound Q', $Q' \models^{\sharp/\flat} Q$ implies $Q \models^{\sharp/\flat} Q'$.

In the following, for any set X, let $min(X) = \{\, x \in X \mid \neg\exists y \in X.\, y \subset x \,\}$ and $max(X) = \{\, x \in X \mid \neg\exists y \in X.\, x \subset y \,\}$. We often denote $min\, X$ and $max\, X$ by omitting (). For two sets of literals $S, T \subseteq Lit$, we define

$$S \uplus T = \begin{cases} S \cup T, & \text{if } S \cup T \text{ does not contain a pair of complementary literals;} \\ Lit, & \text{otherwise.} \end{cases}$$

Theorem 3.1. (1) *An EDP Q is an mub of P_1 and P_2 in $\langle \mathcal{EDP}, \models^\sharp \rangle$ iff*

$$A(Q) = min\{\, S \uplus T \mid S \in A(P_1),\ T \in A(P_2)\,\}.$$

(2) *An EDP Q is an mlb of P_1 and P_2 in $\langle \mathcal{EDP}, \models^\sharp \rangle$ iff*

$$A(Q) = min(A(P_1) \cup A(P_2)).$$

(3) *An EDP Q is an mub of P_1 and P_2 in $\langle \mathcal{EDP}, \models^\flat \rangle$ iff*

$$A(Q) = max(A(P_1) \cup A(P_2)).$$

(4) *An EDP Q is an mlb of P_1 and P_2 in $\langle \mathcal{EDP}, \models^\flat \rangle$ iff*

$$A(Q) = max\{\, S \cap T \mid S \in A(P_1),\ T \in A(P_2)\,\}.$$

Proof. Because of the space limitation, we prove (1) and (2) only, but the proof of (3) and (4) can be constructed in a similar way to that of (2) and (1), respectively.

(1) Q is an upper bound of P_1 and P_2 in $\langle \mathcal{EDP}, \models^\sharp \rangle$ iff $Q \models^\sharp P_1$ and $Q \models^\sharp P_2$
iff $\forall S \in A(Q)\, \exists T_1 \in A(P_1).\, T_1 \subseteq S$ and $\forall S \in A(Q)\, \exists T_2 \in A(P_2).\, T_2 \subseteq S$
iff $\forall S \in A(Q)\, \exists T_1 \in A(P_1)\, \exists T_2 \in A(P_2).\, T_1 \cup T_2 \subseteq S$. (*)

Now, suppose that $A(Q)$ is given as $min\{T_1 \uplus T_2 \mid T_1 \in A(P_1),\ T_2 \in A(P_2)\}$. This Q is an upper bound of P_1 and P_2 because (*) is satisfied. If Q is contradictory, then $A(Q) = \{Lit\}$. Then, for any $T_1 \in A(P_1)$ and any $T_2 \in A(P_2)$, $T_1 \uplus T_2 = Lit$, that is, $T_1 \cup T_2$ is inconsistent. In this case, Q is an mub. Else if Q is incoherent, then $A(Q) = \emptyset$. Then, for any $T_1 \in A(P_1)$ and any $T_2 \in A(P_2)$, $T_1 \uplus T_2$ is undefined, and thus, $A(P_1) = \emptyset$ or $A(P_2) = \emptyset$. That is, either P_1 or P_2 is incoherent. In this case, Q is an mub too.

Next, consider the case that Q is consistent. Suppose further that Q is not an mub. Then, there is $Q' \in \mathcal{EDP}$ such that (i) Q' is an upper bound of P_1 and P_2, (ii) $Q \models^\sharp Q'$, and (iii) $Q' \not\models^\sharp Q$. Here, (ii) and (iii) imply that $A(Q) \neq A(Q')$ by Theorem 2.1. Because $A(Q) \neq \{Lit\}$, it holds that, for any $S \in A(Q)$, $S = T_1 \uplus T_2 = T_1 \cup T_2$ for some $T_1 \in A(P_1)$ and $T_2 \in A(P_2)$. For this S, there is an answer set $S' \in A(Q')$ such that $S' \subseteq S$ by (ii) and that $S' = T_3 \cup T_4$ for some $T_3 \in A(P_1)$ and $T_4 \in A(P_2)$ by (i) and (*). Hence, $T_3 \cup T_4 \subseteq T_1 \cup T_2$. By the minimality of $A(Q)$ with respect to the operation min, it must be $T_3 \cup T_4 = T_1 \cup T_2$, and thus $S' = S$. Hence, $A(Q) \subseteq A(Q')$. By $A(Q) \neq A(Q')$, there is $U \in A(Q')$ such that $U \notin A(Q)$. Again, $U = T' \cup T''$ for some $T' \in A(P_1)$ and $T'' \in A(P_2)$ by (i) and (*). However, there must be some $V \in A(Q)$ such that $V \subseteq U$ by the construction of $A(Q)$ and the minimality of $A(Q)$ with respect to the operation min. Because $U \notin A(Q)$, $V \subset U$ holds. However, by $Q \models^\sharp Q'$, there is $U' \in A(Q')$ such that $U' \subseteq V$ and hence $U' \subset U$. This contradicts the fact that $A(Q')$ is an anti-chain. Therefore, Q is an mub of P_1 and P_2.

(2) Q is a lower bound of P_1 and P_2 in $\langle \mathcal{EDP}, \models^\sharp \rangle$ iff $P_1 \models^\sharp Q$ and $P_2 \models^\sharp Q$
iff $\forall S \in A(P_1)\, \exists T \in A(Q).\, T \subseteq S$ and $\forall S \in A(P_2)\, \exists T \in A(Q).\, T \subseteq S$
iff $\forall S \in A(P_1) \cup A(P_2)\, \exists T \in A(Q).\, T \subseteq S$. (**)

Now, suppose that $A(Q) = min(A(P_1) \cup A(P_2))$. This Q is a lower bound of P_1 and P_2 because (**) is satisfied. If Q is contradictory, then $A(Q) = \{Lit\}$ and both P_1 and P_2 are contradictory. In this case, Q is an mlb. Else if Q is incoherent, then $A(Q) = \emptyset$ and both P_1 and P_2 are incoherent. In this case, Q is an mlb too. Else if Q is consistent, suppose further that Q is not an mlb. Then, there is a lower bound $Q' \in \mathcal{EDP}$ of P_1 and P_2 such that $Q' \models^\sharp Q$ and $A(Q) \neq A(Q')$ by the same argument as the proof of (1). By $Q' \models^\sharp Q$, for any $T' \in A(Q')$, there is $T \in A(Q)$ such that $T \subseteq T'$. By this and the fact that Q' is a lower bound of P_1 and P_2, we have that $\forall S \in A(P_1) \cup A(P_2)\, \exists T' \in A(Q')\, \exists T \in A(Q).\, T \subseteq T' \subseteq S$. By the minimality of $A(Q)$ with respect to the operation min, it must be $T' = T$, and thus $A(Q') \subseteq A(Q)$. By $A(Q) \neq A(Q')$, there is $V \in A(Q)$ such that $V \notin A(Q')$. Since $V \in A(P_1) \cup A(P_2)$ by the construction of $A(Q)$, there must be some $U \in A(Q')$ such that $U \subseteq V$ by (**). Because $V \notin A(Q')$, $U \subset V$ holds. However, by $Q' \models^\sharp Q$, there is $V' \in A(Q)$ such that $V' \subseteq U$ and thus $V' \subset V$. This contradicts the fact that $A(Q)$ is an anti-chain. Therefore, Q is an mlb of P_1 and P_2. □

Example 3.1. Consider P_1, P_2 and P_4 in Example 2.2, where $A(P_1) = \{\{p\}\}$, $A(P_2) = \{\{p\}, \{q\}\}$ and $A(P_4) = \{\{p, q\}\}$. Because $P_4 \models^\sharp P_2$, an mub (resp. mlb) of P_2 and P_4 in $\langle \mathcal{EDP}, \models^\sharp \rangle$ is P_4 (resp. P_2). Correspondingly, $min\{T_1 \uplus T_2 \mid T_1 \in A(P_2),\ T_2 \in A(P_4)\} = min\{\{p, q\}\} = A(P_4)$ and $min(A(P_2) \cup A(P_4)) = min\{\{p\}, \{q\}, \{p, q\}\} = \{\{p\}, \{q\}\} = A(P_2)$. Similarly, an mub (resp. mlb) of P_2 and P_4 in $\langle \mathcal{EDP}, \models^\flat \rangle$ is P_4 (resp. P_2). Correspondingly, $max(A(P_2) \cup A(P_4)) = max\{\{p\}, \{q\}, \{p, q\}\} = \{\{p, q\}\} = A(P_4)$ and $max\{T_1 \cap T_2 \mid T_1 \in A(P_2),\ T_2 \in A(P_4)\} = max\{\{q\}, \{p\}\} = A(P_2)$.

Consider further the program $P_5 = \{\, q \leftarrow not\, p \,\}$, where $A(P_5) = \{\{q\}\}$. Then, P_4 is an mub of P_1 and P_5 in $\langle \mathcal{EDP}, \models^\sharp \rangle$ because $min\{T_1 \uplus T_2 \mid T_1 \in A(P_1),\ T_2 \in A(P_5)\} = min\{\{p, q\}\} = A(P_4)$. Also, P_2 is an mlb of P_1 and P_5 in $\langle \mathcal{EDP}, \models^\sharp \rangle$ and is an mub of P_1 and P_5 in $\langle \mathcal{EDP}, \models^\flat \rangle$ because $min(A(P_1) \cup A(P_5)) = max(A(P_1) \cup A(P_5)) = \{\{p\}, \{q\}\} = A(P_2)$. Finally, $P_6 = \emptyset$ is an mlb of P_1 and P_5 in $\langle \mathcal{EDP}, \models^\flat \rangle$ because $max\{T_1 \cap T_2 \mid T_1 \in A(P_1),\ T_2 \in A(P_5)\} = max\{\emptyset\} = A(P_6)$.

Note that any contradictory program Q is an mub of $\{\, p \leftarrow \,\}$ and $\{\, \neg p \leftarrow \,\}$ because $A(Q) = min\{T_1 \uplus T_2 \mid T_1 = \{p\},\ T_2 = \{\neg p\}\} = min\{Lit\} = \{Lit\}$.

3.2 Lub and Glb on Equivalence Classes

Now, we can construct a poset from the pre-order set $\langle \mathcal{EDP}, \models^{\sharp/\flat} \rangle$ in the usual way as follows. For any program $P \in \mathcal{EDP}$, consider the equivalence class:

$$[P] = \{\, Q \in \mathcal{EDP} \mid A(Q) = A(P) \,\},$$

and then define the relation $\succeq^\sharp$ as:

$$[P] \succeq^\sharp [Q] \quad \text{if} \quad P \models^\sharp Q.$$

We denote the equivalence classes from $\langle \mathcal{EDP}, \models^\sharp \rangle$ as $\mathbf{P}^\sharp$. The relation $\succeq^\flat$ and the equivalence classes $\mathbf{P}^\flat$ are defined in the same way, and we write $\succeq^{\sharp/\flat}$ and

$\mathbf{P}^{\sharp/\flat}$ to represent two cases together. Then, the relation $\succeq^{\sharp/\flat}$ is a partial order on $\mathbf{P}^{\sharp/\flat}$.

Proposition 3.2. *The poset* $\langle \mathbf{P}^{\sharp/\flat}, \succeq^{\sharp/\flat} \rangle$ *constitutes a complete lattice.*

Proof. We prove for $\langle \mathbf{P}^{\sharp}, \succeq^{\sharp} \rangle$. For EDPs P_1 and P_2, consider an EDP P_3 such that $A(P_3) = min\{S \uplus T \mid S \in A(P_1),\ T \in A(P_2)\}$. Then, $[P_3]$ becomes the lub of $[P_1]$ and $[P_2]$ by Theorem 3.1 (1). On the other hand, let P_4 be an EDP such that $A(P_4) = min(A(P_1) \cup A(P_2))$. Then, $[P_4]$ becomes the glb of $[P_1]$ and $[P_2]$ by Theorem 3.1 (2). The top element $\top^{\sharp}$ of $\langle \mathbf{P}^{\sharp}, \succeq^{\sharp} \rangle$ is the class of incoherent EDPs and the bottom element $\bot^{\sharp}$ of $\langle \mathbf{P}^{\sharp}, \succeq^{\sharp} \rangle$ is $[\emptyset]$.

The result for $\langle \mathbf{P}^{\flat}, \succeq^{\flat} \rangle$ can be shown in a similar manner except that the top element $\top^{\flat}$ of $\langle \mathbf{P}^{\flat}, \succeq^{\flat} \rangle$ is the class of contradictory EDPs and the bottom element $\bot^{\flat}$ of $\langle \mathbf{P}^{\flat}, \succeq^{\flat} \rangle$ is the class of incoherent EDPs. □

3.3 Computing Mubs and Mlbs

Theorem 3.1 presents that, given two EDPs P_1 and P_2, there are mubs and mlbs of P_1 and P_2 in $\langle \mathcal{EDP}, \models^{\sharp/\flat} \rangle$. We briefly discuss how to actually construct those EDPs whose answer sets are given as such in a finite domain.

Incidentally, composing programs corresponding to the four cases in Theorem 3.1 has been studied in a series of work by Sakama and Inoue [13,14,15]. In [13], both a program Q such that $A(Q) = A(P_1) \cup A(P_2)$ and a program R such that $A(R) = A(P_1) \cap A(P_2)$ have been composed, where Q is called *generous coordination* of P_1 and P_2 and R is called *rigorous coordination* of P_1 and P_2. Thus, Theorem 3.1 (2) and (3) correspond to generous coordination. On the other hand, [14] produces *composition* of P_1 and P_2, which is a program Q whose answer sets are exactly given as $min\{T_1 \uplus T_2 \mid T_1 \in A(P_1),\ T_2 \in A(P_2)\}$ in Theorem 3.1 (1). The final case (4) in Theorem 3.1 is considered in [15] as *maximal consensus* among P_1 and P_2. The algorithms in [13,14,15] compose such EDPs in time polynomial to the numbers of answer sets and rules in two programs.

There is also a direct and exponential-time algorithm to construct a program that has exactly the given collection of answer sets. Given a set of answer sets $\{S_1, \ldots, S_m\}$, first compute the disjunctive normal form (DNF) $S_1 \vee \cdots \vee S_m$, then convert it into the conjunctive normal form (CNF) $R_1 \wedge \cdots \wedge R_n$. The set of facts $\{R_1 \leftarrow, \ldots, R_n \leftarrow\}$ then has the answer sets $\{S_1, \ldots, S_m\}$. This DNF-CNF transformation produces disjunctive facts only. This is the case even that the given two programs are ELPs, i.e., programs with no disjunction. Technically, the resulting program P is *head-cycle-free*, that is, it contains no positive cycle through disjuncts appearing in the head of a disjunctive rule [1]. Then, P can be converted to an ELP by shifting disjuncts in the head of a rule to the body as NAF-literals in every possible way as leaving one in the head.

4 Generality Relations Relative to Entailment

In traditional studies on generality in first-order clausal theories, the amount of information brought by a program has been measured by the set of logical

formulas entailed by the program. That is, given two monotonic programs P and Q, P is considered more general than Q if P logically entails more formulas than Q [8]. On the other hand, we have defined the two notions of generality for nonmonotonic programs in terms of answer sets. Here, we will connect the generality relations over answer sets with skeptical and credulous entailment in answer set programming. As a result, we will see that our notions of two generality orderings are also reasonable from the viewpoint of entailment relations.

We first review skeptical and credulous entailment in answer set programming.

Definition 4.1. Let P be a program and L a literal. Then, L is a *skeptical consequence* of P if L is included in every answer set of P. L is a *credulous consequence* of P if L is included in some answer set of P. The set of skeptical (resp. credulous) consequences of P is denoted as $skp(P)$ (resp. $crd(P)$).

Proposition 4.1. *If P is a consistent program, then*

$$skp(P) = \bigcap_{S \in A(P)} S, \qquad crd(P) = \bigcup_{S \in A(P)} S.$$

If P is incoherent, then $skp(P) = Lit$ and $crd(P) = \emptyset$. If P is contradictory, then $skp(P) = crd(P) = Lit$.

Example 4.1. Consider P_2 and P_4 in Example 2.2, where $A(P_2) = \{\{p\}, \{q\}\}$ and $A(P_4) = \{\{p, q\}\}$. Then, $crd(P_2) = crd(P_4) = skp(P_4) = \{p, q\}$, and $skp(P_2) = \emptyset$.

The orderings relative to skeptical and credulous entailment relations between two programs are defined as follows.

Definition 4.2. Let P and Q be EDPs. Then, we write:

$$P \models_{skp} Q \ \text{ if } \ skp(Q) \subseteq skp(P),$$
$$P \models_{crd} Q \ \text{ if } \ crd(Q) \subseteq crd(P).$$

We say P is *more general than Q under skeptical entailment* if $P \models_{skp} Q$. Likewise, P is *more general than Q under credulous entailment* if $P \models_{crd} Q$.

For notational convenience, we write $P \models_{s/c} Q$ when distinction between skeptical and credulous entailment is not important.

Proposition 4.2. *The relation $\models_{s/c}$ is a pre-order on $\mathcal{EDP}$.*

As in the case of $\sharp/\flat$-generality relations, the pre-order set $\langle \mathcal{EDP}, \models_{s/c} \rangle$ is turned into a poset as follows. For any program $P \in \mathcal{EDP}$ and the equivalence class

$$[P]_s = \{\, Q \in \mathcal{EDP} \mid P \models_{skp} Q,\ \ Q \models_{skp} P \,\},$$

we define

$$[P]_s \succeq_{skp} [Q]_s \quad \text{if} \quad P \models_{skp} Q,$$

and denote the equivalence classes from $\langle \mathcal{EDP}, \models_{skp} \rangle$ as $\mathbf{P}_{skp}$. The relation $\succeq_{crd}$ and the equivalence classes $\mathbf{P}_{crd}$ are defined in the same way, and we write $\succeq_{s/c}$ and $\mathbf{P}_{s/c}$ to represent two cases together. Then, the relation $\succeq_{s/c}$ is a partial order on $\mathbf{P}_{s/c}$.

Proposition 4.3. *The poset $\langle \mathbf{P}_{s/c}, \succeq_{s/c} \rangle$ constitutes a complete lattice.*

Proof. We prove for $\langle \mathbf{P}_{skp}, \succeq_{skp} \rangle$. The result for $\langle \mathbf{P}_{crd}, \succeq_{crd} \rangle$ is shown in a similar manner. For programs P_1 and P_2, there is a program P_3 such that $skp(P_3) = skp(P_1) \cup skp(P_2)$. (An instance of such a program is $P_3 = \{ L \leftarrow \mid L \in skp(P_1) \cup skp(P_2) \}$.) Then, $[P_3]_s$ becomes the lub of $[P_1]_s$ and $[P_2]_s$. On the other hand, for programs P_1 and P_2, there is a program P_4 such that $skp(P_4) = skp(P_1) \cap skp(P_2)$. Then, $[P_4]_s$ becomes the glb of $[P_1]_s$ and $[P_2]_s$. The top element of $\langle \mathbf{P}_{skp}, \succeq_{skp} \rangle$ is the class of incoherent EDPs and the bottom element of $\langle \mathbf{P}_{skp}, \succeq_{skp} \rangle$ is $[\emptyset]$. □

Now, we relate the $\sharp$- and $\flat$-generality relations with the generality relations under skeptical and credulous entailment.

Theorem 4.1. *Let P and Q be EDPs. Then, the following two hold.*

(1) *If $P \models^{\sharp} Q$ then $P \models_{skp} Q$.*
(2) *If $P \models^{\flat} Q$ then $P \models_{crd} Q$.*

Proof. (1) Assume that $P \models^{\sharp} Q$. If P is inconsistent, then $skp(P) = Lit$ and thus $P \models_{skp} Q$. Suppose that P is consistent and $L \in skp(Q)$. Then, $L \in T$ for every answer set $T \in A(Q)$. By $P \models^{\sharp} Q$, for any $S \in A(P)$, there is an answer set $T' \in A(Q)$ such that $T' \subseteq S$. Since $L \in T'$, $L \in S$ too. That is, $L \in skp(P)$. Hence, $P \models_{skp} Q$.

(2) Assume that $P \models^{\flat} Q$. If P is incoherent, then P is in the bottom element of $\langle \mathbf{P}^{\flat}, \succeq^{\flat} \rangle$, and hence Q is too. Then, $crd(P) = crd(Q) = \emptyset$ and thus $P \models_{crd} Q$. Else if P is contradictory, then $crd(P) = Lit$ and thus $P \models_{crd} Q$. Suppose that P is consistent and $L \in crd(Q)$. Then, $L \in T$ for some answer set $T \in A(Q)$. By $P \models^{\flat} Q$, there is an answer set $S \in A(P)$ such that $T \subseteq S$. Hence, $L \in S$ and thus $L \in crd(P)$. That is, $P \models_{crd} Q$. □

Theorem 4.1 tells us that, (1) the more $\sharp$-general a program is, the more it entails skeptically, and that (2) the more $\flat$-general a program is, the more it entails credulously. That is, the Smyth and Hoare orderings over programs reflect the amount of information by skeptical and credulous entailment, respectively. The converse of each property in Theorem 4.1 does not hold in general.

Example 4.2. For Example 4.1, $P_4 \models^{\sharp} P_2$ and $P_4 \models^{\flat} P_2$. Correspondingly, $skp(P_2) \subset skp(P_4)$ and $crd(P_2) = crd(P_4)$, which verify Theorem 4.1.

On the other hand, $crd(P_2) = crd(P_4)$ also implies $P_2 \models_{crd} P_4$, but $P_2 \not\models^{\flat} P_4$. Similarly, for the program $P_6 = \emptyset$, we have $skp(P_6) = \emptyset = skp(P_2)$. Then, $P_6 \models_{skp} P_2$, but $P_6 \not\models^{\sharp} P_2$.

By Theorem 4.1, the relation $\models^{\sharp/\flat}$ is a *refinement* of the relation $\models_{s/c}$, respectively. Comparing these two kinds of ordering relations, we claim that $\models^{\sharp/\flat}$ is more useful than $\models_{s/c}$ as generality criteria. This is because each equivalence class $[P] \in \mathbf{P}^{\sharp/\flat}$ is the set of programs which are weakly equivalent to P (Theorem 2.1), although there is no such a simple property for the equivalence classes $\mathbf{P}_{s/c}$. For Example 4.2, $crd(P_2) = crd(P_4)$ but $A(P_2) \neq A(P_4)$, and $skp(P_2) = skp(P_6)$ but $A(P_2) \neq A(P_6)$.

The next theorem presents interesting relationships between mubs/mlbs under the generality $\models^{\sharp/\flat}$ and skeptical/credulous entailment.

Lemma 4.1. *Let P_1 and P_2 be EDPs.*

(1) *If Q is an mub of P_1 and P_2 in $\langle \mathcal{EDP}, \models^{\sharp} \rangle$ then $skp(Q) = skp(P_1) \uplus skp(P_2)$.*
(2) *If Q is an mlb of P_1 and P_2 in $\langle \mathcal{EDP}, \models^{\sharp} \rangle$ then $skp(Q) = skp(P_1) \cap skp(P_2)$.*
(3) *If Q is an mub of P_1 and P_2 in $\langle \mathcal{EDP}, \models^{\flat} \rangle$ then $crd(Q) = crd(P_1) \cup crd(P_2)$.*
(4) *If Q is an mlb of P_1 and P_2 in $\langle \mathcal{EDP}, \models^{\flat} \rangle$ then $crd(Q) = crd(P_1) \cap crd(P_2)$.*

Proof. An mub/mlb of two programs under each ordering is given by Theorem 3.1. Then, (1) can be proved by [14, Proposition 3.5(2)], (2) can be proved by [13, Proposition 3.1-1(b)], (3) can be proved by [13, Proposition 3.1-1(a)], and (4) can be proved by [15, Proposition 5(3)]. □

Theorem 4.2. *Let P_1 and P_2 be EDPs.*

(1) *An mub of P_1 and P_2 in $\langle \mathcal{EDP}, \models^{\sharp} \rangle$ is an mub of P_1 and P_2 in $\langle \mathcal{EDP}, \models_{skp} \rangle$.*
(2) *An mlb of P_1 and P_2 in $\langle \mathcal{EDP}, \models^{\sharp} \rangle$ is an mlb of P_1 and P_2 in $\langle \mathcal{EDP}, \models_{skp} \rangle$.*
(3) *An mub of P_1 and P_2 in $\langle \mathcal{EDP}, \models^{\flat} \rangle$ is an mub of P_1 and P_2 in $\langle \mathcal{EDP}, \models_{crd} \rangle$.*
(4) *An mlb of P_1 and P_2 in $\langle \mathcal{EDP}, \models^{\flat} \rangle$ is an mlb of P_1 and P_2 in $\langle \mathcal{EDP}, \models_{crd} \rangle$.*

Proof. Each mub/mlb of P_1 and P_2 in $\langle \mathcal{EDP}, \models_{s/c} \rangle$ satisfies each equation (1) to (4) in Lemma 4.1. Then each property holds by Lemma 4.1. □

5 Strong Generality Relations over Logic Programs

In the previous sections, we have seen that the relation $\models^{\sharp/\flat}$ is useful for determining the degree of generality of EDPs. However, because $\sharp/\flat$-generality is determined solely by the answer sets of each program, sometimes the criteria is not suitable for applications in dynamic domains. For example, for ELPs $P = \{ p \leftarrow not\ q \}$ and $Q = \{ p \leftarrow q \}$, we have $P \models^{\sharp} Q$. Then, adding $R = \{ q \leftarrow \}$ to both programs makes the results in reverse order, i.e., $Q \cup R \models^{\sharp} P \cup R$. In this section, we will thus introduce context-sensitive notions of generality.

Definition 5.1. Let P and Q be programs. P is *strongly more $\sharp$-general than Q* (written $P \trianglerighteq^{\sharp} Q$) if $P \cup R \models^{\sharp} Q \cup R$ for any program R. Similarly, P is *strongly more $\flat$-general than Q* (written $P \trianglerighteq^{\flat} Q$) if $P \cup R \models^{\flat} Q \cup R$ for any program R.

We write $\trianglerighteq^{\sharp/\flat}$ to represent both $\trianglerighteq^{\sharp}$ and $\trianglerighteq^{\flat}$ together. It is easy to see that strong $\sharp/\flat$-generality implies $\sharp/\flat$-generality.

Proposition 5.1. *Let P and Q be EDPs. If $P \trianglerighteq^{\sharp/\flat} Q$ then $P \models^{\sharp/\flat} Q$.*

Strong $\sharp/\flat$-generality can be contrasted with the notion of strong equivalence [6] in answer set programming. In fact, we have the following correspondence between strong generality and strong equivalence.

Theorem 5.1. *Let P and Q be EDPs. Then, the following three are equivalent:*

(1) $P \trianglerighteq^{\sharp} Q$ *and* $Q \trianglerighteq^{\sharp} P$;
(2) $P \trianglerighteq^{\flat} Q$ *and* $Q \trianglerighteq^{\flat} P$;
(3) P *and* Q *are strongly equivalent.*

Proof. $P \trianglerighteq^{\sharp/\flat} Q$ and $Q \trianglerighteq^{\sharp/\flat} P$
iff $P \cup R \models^{\sharp/\flat} Q \cup R$ and $Q \cup R \models^{\sharp/\flat} P \cup R$ for any program R
iff $P \cup R$ and $Q \cup R$ are weakly equivalent for any program R (by Theorem 2.1)
iff P and Q are strongly equivalent. □

Example 5.1. Consider the four EDPs in Example 2.2. Then, $P_1 \trianglerighteq^{\sharp} P_2 \trianglerighteq^{\sharp} P_3$ holds. However, $P_4 \not\trianglerighteq^{\sharp} P_1$ (take $R = \{ q \leftarrow p \}$ then $P_1 \cup R$ is incoherent while $P_4 \cup R = P_4$, hence $P_4 \cup R \not\models^{\sharp} P_1 \cup R$), $P_4 \not\trianglerighteq^{\sharp} P_2$ (take $R' = \{ q \leftarrow p, \quad p \leftarrow q \}$ then $P_2 \cup R'$ is incoherent while $P_4 \cup R' = P_4$, hence $P_4 \cup R' \not\models^{\sharp} P_2 \cup R'$), $P_3 \not\trianglerighteq^{\sharp} P_2$ (take R' above then $P_2 \cup R'$ is incoherent while $P_3 \cup R'$ is consistent, hence $P_3 \cup R' \not\models^{\sharp} P_2 \cup R'$), and $P_4 \not\trianglerighteq^{\sharp} P_3$ (take $R'' = \{ \leftarrow not\, p, \quad \leftarrow not\, q \}$ then $P_3 \cup R''$ is incoherent while $P_4 \cup R''$ is consistent, hence $P_4 \cup R'' \not\models^{\sharp} P_3 \cup R''$).

On the other hand, $P_3 \trianglerighteq^{\flat} P_2 \trianglerighteq^{\flat} P_1$ holds under the relation $\trianglerighteq^{\flat}$.

In Example 5.1, the two weakly equivalent programs P_2 and P_3 are not strongly equivalent, and then $P_2 \trianglerighteq^{\sharp} P_3$ but $P_3 \not\trianglerighteq^{\sharp} P_2$. This fact can be intuitively explained as follows. $P_2 = \{ p \leftarrow not\, q, \quad q \leftarrow not\, p \}$ is more informative than $P_3 = \{ p; q \leftarrow \ \}$ in the sense that the derivation of p (or q) depends on the absence of q (or p) in P_2. However, no such information is obtained in P_3 so that we have a chance to extend the contents by adding $R' = \{ q \leftarrow p, \quad p \leftarrow q \}$ to P_3, which is impossible for P_2. On the other hand, under the relation $\trianglerighteq^{\flat}$, we have $P_3 \trianglerighteq^{\flat} P_2$ but $P_2 \not\trianglerighteq^{\flat} P_3$. This is because any incoherent program becomes a top element $\top^{\sharp}$ under $\trianglerighteq^{\sharp}$, while it is a bottom element $\bot^{\flat}$ under $\trianglerighteq^{\flat}$. In this regard, the next proposition gives a necessary condition for strong generality.

Proposition 5.2. *Let P and Q be EDPs.*

(1) *If $P \trianglerighteq^{\sharp} Q$ then $A(Q \cup R) = \emptyset$ implies $A(P \cup R) = \emptyset$ for any EDP R.*
(2) *If $P \trianglerighteq^{\flat} Q$ then $A(P \cup R) = \emptyset$ implies $A(Q \cup R) = \emptyset$ for any EDP R.*

Proposition 5.3. *Both $\langle \mathcal{EDP}, \trianglerighteq^{\sharp} \rangle$ and $\langle \mathcal{EDP}, \trianglerighteq^{\flat} \rangle$ are pre-ordered sets.*

As in the case of $\sharp/\flat$-generality relations, from the pre-order set $\langle \mathcal{EDP}, \trianglerighteq^{\sharp/\flat} \rangle$, a poset can be induced over the equivalence classes as usual. This poset also constitutes a complete lattice, but we omit the detail in this paper.

6 Discussion

Theories of generality relations over first-order clauses have been studied in the field of *inductive logic programming* [8,9,10]. These studies mainly focus on the generality relationship between two individual clauses and a generality order is introduced over a set of clauses. By contrast, we considered generality relations between *programs*. Moreover, the main contribution of this paper is a theory of generality relations in *nonmonotonic logic programs*, which contain incomplete information. The generality theory developed in this paper is useful for comparing the amount of information between such programs. To our best knowledge, there has never been a study on generality relations over nonmonotonic logic programs except [12]. Sakama [12] introduces an ordering over default theories and nonmonotonic logic programs. He orders ELPs based on a ten-valued logic, which is different from the domain-theoretic appraoch in this paper.

Computing mubs and mlbs of two programs in this paper is closely related to coordination, composition and consensus in *multi-agent systems*, which have been studied by Sakama and Inoue [13,14,15] (see Section 3.3). *Coordination* [13] is realized by accomodating different beliefs of individual agents. This is done by collecting answer sets of each program. On the other hand, *composition* [14] is realized by merging different answer sets of each programs, and *consensus* [15] is realized by extracting common beliefs from different answer sets of each program. The results of this paper indicate that our generalization theory can serve as a theoretical ground for formalizing social behavior of multiple agents.

The Smyth and Hoare orderings were proposed in domain theory, which is concerned with mathematical structures to formalize the denotational semantics of programming languages [11,16,4]. The recursive, concurrent and nondeterministic nature of programming constructs have been modeled on these order-theoretic powerdomains. In this viewpoint, answer set programming also imposes non-determinism, and we thus regard that domain theory is suitable to analyze structural properties of answer sets. However, there is only a few work on domain-theoretic foundations on logic programming, and in particular, no one has proposed a domain-theoretic method to compare the amount of information brought by a logic program. Zhang and Rounds [17] represent the semantics of disjunctive logic programs on Scott's information systems using Smyth powerdomain. In contrast to our work, Zhang and Rounds are concerned with the semantics of individual programs, and do not consider comparison of multiple programs in powerdomains. Eiter *et al.* [2] have proposed a framework for comparing programs with respect to binary relations on powersets of the *projected* answer sets of the programs, but relations in their framework are limited to equivalence and inclusion, and generality is not taken into account.

Finally, it should be noted that our framework to compare programs in Section 2.2 is farily general, so that its applicability is not only limited to answer set programming. In fact, $A(P)$ for a program P in Definition 2.2 can be given by any semantics of P as long as it is defined as a subset of Lit.

Several issues are left open. Because we have developed the theory of generality relations from the semantical viewpoint, exploring the syntactical counter-

part is an important future topic. For example, construction of a more (or less) (strongly) ♯/♭-general program of a given program by a syntactical manipulation is useful for generalizing or specializing a program in inductive logic programming. For another issue, strong generality in the current form seems too strong, and it must be meaningful to relax its context-dependent generality condition from one with respect to all EDPs to one with respect to a subclass of EDPs.

References

1. R. Ben-Eliyahu and R. Dechter. Propositional semantics for disjunctive logic programs. *Annals of Mathematics and Artificial Intelligence*, 12(1):53–87, 1994.
2. T. Eiter, H. Tompits, and S. Woltran. On solution correspondences in answer-set programming. In: *Proceedings of the 19th International Joint Conference on Artificial Intelligence*, pp. 97–102, 2005.
3. M. Gelfond and V. Lifschitz. Classical negation in logic programs and disjunctive databases. *New Generation Computing*, 9:365–385, 1991.
4. C. A. Gunter and D. S. Scott. Semantic Domains. In: J. van Leeuwen (ed.), *Handbook of Theoretical Computer Science*, Vol. B, pp. 633–674, North-Holland, 1990.
5. K. Inoue and C. Sakama. Disjunctive explanations. In: *Proceedings of the 18th International Conference on Logic Programming, Lecture Notes in Computer Science*, 2401, pp. 317–332, Springer, 2002.
6. V. Lifschitz, D. Pearce and A. Valverde. Strongly equivalent logic programs. *ACM Transactions on Computational Logic*, 2:526–541, 2001.
7. M. J. Maher. Equivalence of logic programs. In: J. Minker (ed.), *Foundations of Deductive Databases and Logic Programming*, pp. 627–658, Morgan Kaufmann, 1988.
8. T. Niblett. A study of generalization in logic programs. In: *Proceedings of the 3rd European Working Sessions on Learning (EWSL-88)*, pp. 131–138, Pitman, 1988.
9. S.-H. Nienhuys-Cheng and R. de Wolf. *Foundations of inductive logic programming. Lecture Notes in Artificial Intelligence*, 1228, Springer, 1997.
10. G. D. Plotkin. A note on inductive generalization. In: B. Meltzer and D. Michie (eds.), *Machine Intelligence* 5, pp. 153–63, Edinburgh University Press, 1970.
11. G. D. Plotkin. A powerdomain construction. *SIAM Journal of Computing*, 5:452–287, 1976.
12. C. Sakama. Ordering default theories and nonmonotonic logic programs. *Theoretical Computer Science*, 338:127–152, 2005.
13. C. Sakama and K. Inoue. Coordination between logical agents. In: *Post-proceedings of the 5th International Workshop on Computational Logic in Multi-Agent Systems, Lecture Notes in Artificial Intelligence*, 3487, pp. 161-177, Springer, 2005.
14. C. Sakama and K. Inoue. Combining answer sets of nonmonotonic logic programs. In: *Post-proceedings of the 6th International Workshop on Computational Logic in Multi-Agent Systems, Lecture Notes in Artificial Intelligence*, 3900, pp. 320-339, Springer, 2006.
15. C. Sakama and K. Inoue. Constructing consensus logic programs. Submitted for publication, 2006.
16. M. B. Smyth. Power domains. *Journal of Computer and System Sciences*, 16:23–36, 1978.
17. G. Q. Zhang and W. C. Rounds. Semantics of logic programs and representation of Smyth powerdomain. In: K. Keimel *et al* (eds.), *Proceedings of the 1st International Symposium on Domain Theory*, pp. 151–181, Kluwer, 2001.