Exploring Relations between Answer Set Programs*

Katsumi Inoue[1] and Chiaki Sakama[2]

[1] National Institute of Informatics
2-1-2 Hitotsubashi, Chiyoda-ku, Tokyo 101-8430, Japan
[2] Department of Computer and Communication Sciences, Wakayama University
Sakaedani, Wakayama 640-8510, Japan

Abstract. Equivalence and generality relations over logic programs have been proposed in answer set programming to semantically compare information contents of logic programs. In this paper, we overview previous relations of answer set programs, and propose a general framework that subsumes previous relations. The proposed framework allows us to compare programs possibly having non-minimal answer sets as well as to explore new relations between programs. Such new relations include relativized variants of generality relations over logic programs. By selecting contexts for comparison, the proposed framework can represent weak, strong and uniform variants of generality, inclusion and equivalence relations. These new relations can be applied to comparison of abductive logic programs and coordination of multiple answer set programs.

1 Introduction

Relations between theories have attracted a lot of interests in design and maintenance of knowledge bases. In particular, the notion of *equivalence* is important for verification, simplification and optimization of logic programs. Maher [26] introduced the notion of *equivalence as program segments* for definite programs, which has later been explored as *strong equivalence* for answer set programming by Lifschitz *et al.* [21]. Sagiv [37] defined the notion of *uniform equivalence* of Datalog programs as equivalence of the outputs of programs for any set of input atoms, which has also been extended for answer set programming by Eiter and Fink [4]. These two notions of equivalence are then unified in a more general concept called *relative/relativized equivalence* [25, 15, 49, 7, 31, 50, 48], which characterizes equivalence with respect to *contexts*. By changing contexts, we can consider various forms of equivalence relations in answer set programming.

Inter-theory relations between logic programs are not necessarily limited to equivalence. We also would like to know what kind of relations hold between programs when they are not equivalent. Eiter *et al.* [7] define a general correspondence framework to compare answer sets of programs not only under

* This research is supported in part by the 2008-2011 JSPS Grant-in-Aid for Scientific Research (A) No. 20240016.

equivalence but under *inclusion*, and Oetsch *et al.* [31] extend the framework to the uniform variant. Inoue and Sakama [17] define another inter-theory relation called *generality* for answer set programming. Their main concern is to compare the amounts of information brought by logic programs and thus to assess the relative value of each program. It turns out that inclusion by [7] is a special case of generality.

Generality relations have many important applications. First, generality has been central in the theory of *inductive logic programming* (ILP) [34, 28, 29] as a criterion to rank hypotheses whenever alternative hypotheses can be considered. Intuitively, a program P_1 is considered *more general* than a program P_2 if P_1 brings more information than P_2. In *learning nonmonotonic theories* [13, 32, 39, 45], hypotheses are constructed in the form of logic programs with negation as failure. Then, we need to rank such hypotheses according to some generality relations. Secondly, answer set programming is used for describing and refining knowledge bases related to *ontologies* [6]. In this process, generality relations can be used as criteria to judge whether descriptions are really refined [18]. Thirdly, integration, merging, and coordination of multiple descriptions are important in *multi-agent systems*; one may modify her own description in accordance with those from other agents by constructing a *minimal generalization* or a *maximal specialization* of multiple descriptions [44], or may *compose a new description* by gathering agents' descriptions [42], or may build a *consensus* between agents' beliefs [43]. These combination operations can be formalized as *minimal upper* or *maximal lower bounds* of multiple programs under generality relations [17].

In this paper, we will consider stronger versions of generality. The idea is to introduce some focused contexts for comparison, which makes two programs comparable under particular program extensions. As in the case of equivalence, *strong* or *uniform* variants of generality is considered to represent a situation that a generality relation between two programs is preserved after adding any set of rules or facts, respectively. Moreover, these two notions of generality can be unified into the notion of *relativized generality* as in the case of relativized equivalence. Motivation behind this relativized generality lies in the fact that we learn from it *robustness* of generality between programs: one program brings more information than another whatever rules in a focused context are added to those programs. More justifications for consideration of relativized generality will be discussed from the viewpoint of *inductive generality*. As an application of relativized generality, we will show that explanatory power of *abductive logic programs* [20] can be compared using the proposed framework. Although the notion of *abductive generality* is defined in [19] from the viewpoint of abductive explanations, we will see that it can be characterized as a special case of relativized generality. Further applications of this generality would be logical foundations of coordination between multiple *abductive agents*.

Note that the generality relations are defined in [17] for *extended disjunctive programs* using the Smyth ($\sharp$) and Hoare ($\flat$) orderings from domain theory [11], which respectively reflect orderings on inclusion relations of the *skeptical* and *credulous* consequences. Those previous results require an anti-chain property

for the answer sets of programs, which limits the applicability of the framework to EDPs. In this paper, these previous orderings as well as newly proposed orderings can be applied to more extended classes of programs that may possess non-minimal answer sets [24, 40, 14, 23, 8], as well as logic programs with aggregates [46, 27, 47]. Abductive logic programs [20] can also be considered to have non-minimal answer sets due to augmentation with abducibles even when the background program is an EDP.

The rest of this paper is organized as follows. Section 2 defines $\sharp$- and $\flat$-generality relations over the class of all sets of literal sets. Section 3 applies these generality relations to ordering logic programs. Section 4 considers strong, uniform and relativized variants of generality relations and analyzes their properties. Section 5 relates relativized generality with abductive generality. Section 6 gives related work and Section 7 summarizes the paper.

2 Generality Relations over Semantic Structures

This section presents generality relations defined over the sets of literal sets. We recall some mathematical definitions about domains [11]. A *pre-order* (or *quasi-order*) $\succcurlyeq$ is a binary relation which is reflexive and transitive. A pre-order $\succcurlyeq$ 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) $\succcurlyeq$ on D. For a pre-ordered set $\langle D, \succcurlyeq \rangle$ and $x, y \in D$, we write $x \succ y$ if $x \succcurlyeq y$ and $y \not\succcurlyeq x$.

For a pre-ordered set $\langle D, \succcurlyeq \rangle$ and any set $X \subseteq D$, we denote the minimal and maximal elements of X as follows.

$$min_{\succcurlyeq}(X) = \{\, x \in X \mid \neg\exists y \in X.\, x \succ y \,\},$$
$$max_{\succcurlyeq}(X) = \{\, x \in X \mid \neg\exists y \in X.\, y \succ x \,\}.$$

We often denote these as $min(X)$ and $max(X)$ by omitting $\succcurlyeq$. We also assume that the relation $\succcurlyeq$ is well-founded (resp. upwards well-founded) on D[3] whenever $min_{\succcurlyeq}(X)$ (resp. $max_{\succcurlyeq}(X)$) is concerned in order to guarantee the existence of a minimal (resp. maximal) element of any $X \subseteq D$.[4] Note that, when D is finite, any pre-order is both well-founded and upwards well-founded on D.

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

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

and the *Hoare order* is defined as

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

[3] A relation R is *well-founded* on a class D iff every non-empty subset of D has a minimal element with respect to R. A relation R is *upwards well-founded* on D iff the inverse relation R^{-1} is well-founded on D.

[4] For example, Theorems 2.1 (2), 2.2 (2) and 3.2 (2) assume the upwards well-foundedness. Note that the relation $\subseteq$ is well-founded on D even if D is infinite.

The relations $\succeq^\sharp$ and $\succeq^\flat$ are pre-orders on $\mathcal{P}(D)$, and both $\langle \mathcal{P}(D), \succeq^\sharp \rangle$ and $\langle \mathcal{P}(D), \succeq^\flat \rangle$ are pre-ordered sets. Note that the orderings $\succeq^\sharp$ and $\succeq^\flat$ are slightly different from those in domain theory:[5] we allow the empty set $\emptyset$ ($\in \mathcal{P}(D)$) as both the top element $\top^\sharp$ in $\langle \mathcal{P}(D), \succeq^\sharp \rangle$ and the bottom element $\bot^\flat$ in $\langle \mathcal{P}(D), \succeq^\flat \rangle$.

Example 2.1 Consider the poset $\langle \mathcal{P}(\{p, q\}), \supseteq \rangle$. Then, we have $\{\{p, q\}\} \succeq^\sharp \{\{p\}\} \succeq^\sharp \{\{p\}, \{q\}\}$. On the other hand, $\{\{p, q\}\} \succeq^\flat \{\{p\}, \{q\}\} \succeq^\flat \{\{p\}\}$. Since $\{\emptyset, \{p\}\} \succeq^\sharp \{\emptyset, \{q\}\} \succeq^\sharp \{\emptyset, \{p\}\}$ holds, $\succeq^\sharp$ is not a partial order.

We now define orderings over the sets of literal sets. We assume a first-order language, and denote by *Lit* the set of all ground literals in the language. A literal set $T \subseteq Lit$ is *inconsistent* if T contains a pair of complementary literals $L, \neg L$; otherwise, T is *consistent*. The *(logical) closure* of T is defined as

$$cl(T) = \begin{cases} T\,, & \text{if } T \text{ is consistent;} \\ Lit\,, & \text{if } T \text{ is inconsistent.} \end{cases}$$

A literal set T is *closed* if $T = cl(T)$ holds.

We define a semantic structure called a *composite set* as an element in $\mathcal{P}(\mathcal{P}(Lit))$, i.e., a class of sets of ground literals from *Lit*. A composite set is *consistent* if it contains a consistent literal set; otherwise, it is *inconsistent*. Hence, if a composite set Σ is inconsistent, then either $\Sigma = \emptyset$ or every set $T \in \Sigma$ is inconsistent. We denote the closures of a composite set Σ as $Cl(\Sigma) = \{cl(T) \mid T \in \Sigma\}$. Two composite sets Σ_1 and Σ_2 are *equivalent*, denoted as $\Sigma_1 \equiv \Sigma_2$, if $Cl(\Sigma_1) = Cl(\Sigma_2)$. A composite set Σ is *closed* if $\Sigma = Cl(\Sigma)$ holds. Given a pre-order $\succcurlyeq$, a composite set Σ is *irredundant (with respect to $\succcurlyeq$)* if Σ is an anti-chain on the set $\langle \mathcal{P}(Lit), \succcurlyeq \rangle$, that is, for any $S, T \in \Sigma$, $S \succcurlyeq T$ implies $T \succcurlyeq S$.

To define generality ordering over composite sets, let us assume a pre-ordered set $\langle D, \succcurlyeq \rangle$ such that the domain D is $\mathcal{P}(Lit)$, i.e., the class of sets of ground literals in the language, and the pre-order $\succcurlyeq$ is the *inclusion* relation $\supseteq$ over $\mathcal{P}(Lit)$. We denote by $\mathcal{LS}$ the class of all composite sets which can be constructed in the language. That is, $\mathcal{LS} = \mathcal{P}(\mathcal{P}(Lit))$. Then, the Smyth and Hoare orderings on $\mathcal{P}(\mathcal{P}(Lit))$ can be defined, which enable us to order composite sets.

Definition 2.1 Assume a pre-ordered set $\langle \mathcal{P}(Lit), \supseteq \rangle$, and let Σ_1 and Σ_2 be any two composite sets in $\mathcal{LS}$. Σ_1 is *more $\sharp$-general than (or equal to)* Σ_2, written $\Sigma_1 \models^\sharp \Sigma_2$, if $Cl(\Sigma_1) \succeq^\sharp Cl(\Sigma_2)$. Σ_1 is *more $\flat$-general than (or equal to)* Σ_2, written $\Sigma_1 \models^\flat \Sigma_2$, if $Cl(\Sigma_1) \succeq^\flat Cl(\Sigma_2)$.

By definition, $\sharp$- and $\flat$-generality reflect the following situations. $\Sigma_1 \models^\sharp \Sigma_2$ means that any set in Σ_1 has logically more information than (or equal to) some set in Σ_2. On the other hand, $\Sigma_1 \models^\flat \Sigma_2$ means that any set in Σ_2 has logically less information than (or equal to) some set in Σ_1. For notational convenience,

[5] This is because we enable comparison of all classes of programs by associating $\emptyset$ with the class of programs having no answer set [17].

we write $\models^{\sharp/\flat}$ to represent the $\sharp$- or $\flat$-generality relations together, which denotes either $\models^{\sharp}$ or $\models^{\flat}$. It is easy to see that $\langle \mathcal{LS}, \models^{\sharp/\flat} \rangle$ is a pre-ordered set.

In the following theorems, we assume a pre-ordered set $\langle \mathcal{P}(Lit), \supseteq \rangle$, and let $\Sigma_1, \Sigma_2 \in \mathcal{LS}$. The next theorem shows that the minimal (resp. maximal) elements determine the $\models^{\sharp}$ (resp. $\models^{\flat}$) relation between composite sets.

Theorem 2.1 (1) $\Sigma_1 \models^{\sharp} \Sigma_2$ *iff* $min(\Sigma_1) \models^{\sharp} min(\Sigma_2)$.
(2) $\Sigma_1 \models^{\flat} \Sigma_2$ *iff* $max(\Sigma_1) \models^{\flat} max(\Sigma_2)$.

Proof. We prove (1) but (2) can be proved in the same way. Suppose that $\Sigma_1 \models^{\sharp} \Sigma_2$. Then, $\forall S_1 \in min(Cl(\Sigma_1))$, $\exists S_2 \in Cl(\Sigma_2)$ such that $S_1 \supseteq S_2$, and then $\exists S_2' \in min(Cl(\Sigma_2))$ such that $S_2 \supseteq S_2'$. Hence, $min(\Sigma_1) \models^{\sharp} min(\Sigma_2)$. Conversely, suppose that $min(\Sigma_1) \models^{\sharp} min(\Sigma_2)$. For any $T_1 \in Cl(\Sigma_1)$, $\exists T_1' \in min(Cl(\Sigma_1))$ such that $T_1 \supseteq T_1'$. Then by the supposition, for T_1', $\exists T_2 \in min(Cl(\Sigma_2))$ such that $T_1' \supseteq T_2$. Then, $T_2 \in Cl(\Sigma_2)$ by the definition of min. Hence, $\Sigma_1 \models^{\sharp} \Sigma_2$. □

Theorem 2.2 (1) $\Sigma_1 \models^{\sharp} \Sigma_2$ *and* $\Sigma_2 \models^{\sharp} \Sigma_1$ *iff* $min(\Sigma_1) \equiv min(\Sigma_2)$.
(2) $\Sigma_1 \models^{\flat} \Sigma_2$ *and* $\Sigma_2 \models^{\flat} \Sigma_1$ *iff* $max(\Sigma_1) \equiv max(\Sigma_2)$.

Proof. (1) By Theorem 2.1 (1), $\Sigma_1 \models^{\sharp} \Sigma_2$ and $\Sigma_2 \models^{\sharp} \Sigma_1$ iff (i) $min(\Sigma_1) \models^{\sharp} min(\Sigma_2)$ and (ii) $min(\Sigma_2) \models^{\sharp} min(\Sigma_1)$. By (i), $\forall S_1 \in Cl(min(\Sigma_1))$, $\exists S_2 \in Cl(min(\Sigma_2))$ such that $S_1 \supseteq S_2$. Then by (ii), $\exists S_1' \in Cl(min(\Sigma_1))$ such that $S_2 \supseteq S_1'$. Then, $S_1 \supseteq S_1'$ holds, but both belong to $Cl(min(\Sigma_1)) = min(Cl(\Sigma_1))$. Hence, $S_1 = S_1' = S_2$. That is, $S_1 \in Cl(min(\Sigma_1))$ implies $S_1 \in Cl(min(\Sigma_2))$. Hence, $Cl(min(\Sigma_1)) \subseteq Cl(min(\Sigma_2))$. Similarly, $Cl(min(\Sigma_2)) \subseteq Cl(min(\Sigma_1))$.

(2) can also be proved in a similar way. □

Corollary 2.3 *For irredundant composite sets* Σ_1 *and* Σ_2*, the following three are equivalent:* (i) $\Sigma_1 \models^{\sharp} \Sigma_2 \models^{\sharp} \Sigma_1$*;* (ii) $\Sigma_1 \models^{\flat} \Sigma_2 \models^{\flat} \Sigma_1$*;* (iii) $\Sigma_1 \equiv \Sigma_2$.

Proof. If Σ is an irredundant composite set, $max(\Sigma) = min(\Sigma) = \Sigma$ holds. Then the corollary holds by Theorem 2.2. □

The next property states that $\sharp$- and $\flat$-generality relations are *monotonic* with respect to addition of literal sets.

Proposition 2.4 *If* $\Sigma_1 \supseteq \Sigma_2$*, then* $\Sigma_2 \models^{\sharp} \Sigma_1$ *and* $\Sigma_1 \models^{\flat} \Sigma_2$.

In the following propositions, we assume a pre-ordered set $\langle \mathcal{P}(Lit), \supseteq \rangle$.

Proposition 2.5 (Independence) *Let* Σ_1 *and* Σ_2 *be composite sets, and* T_1 *and* T_2 *literal sets. If* $\Sigma_1 \models^{\sharp/\flat} \Sigma_2$ *and* $T_1 \supseteq T_2$*, then* $\Sigma_1 \cup \{T_1\} \models^{\sharp/\flat} \Sigma_2 \cup \{T_2\}$.

Corollary 2.6 (Refinement) *Let* Σ *be a composite set, and* T *a literal set such that* $T \in \Sigma$*. If* S *is a literal set such that* $S \supseteq T$*, then* $(\Sigma \setminus \{T\}) \cup \{S\} \models^{\sharp/\flat} \Sigma$.

Proposition 2.7 (Robustness) *Let* Σ_1*,* Σ_2 *and* Σ_3 *be composite sets. If* $\Sigma_1 \models^{\sharp/\flat} \Sigma_2$ *and* $\Sigma_1 \models^{\sharp/\flat} \Sigma_3$*, then* $\Sigma_1 \models^{\sharp/\flat} \Sigma_2 \cup \Sigma_3$.

Proposition 2.8 (Extended Equivalence) *Let* Σ_1*,* Σ_2*,* Π_1 *and* Π_2 *be composite sets. If* $\Sigma_1 \equiv \Sigma_2$ *and* $\Pi_1 \equiv \Pi_2$*, then* $\Sigma_1 \cup \Pi_1 \equiv \Sigma_2 \cup \Pi_2$.

3 Ordering Logic Programs

This section applies the Smyth and Hoare orderings over $\mathcal{LS}$ to order logic programs. Generalizing the results for the class of extended disjunctive programs [17], we allow any class of programs possibly having non-minimal answer sets.

In this paper, we can compare any class of logic programs as long as the semantics of a program is defined as a *closed composite set*, which is a collection of literal sets called *answer sets* [10] or *stable models* [9] of the program.[6] For example, here we can consider a *general extended disjunctive program* (GEDP) [24, 14], which is a set of *rules* of the form:

$$L_1 ; \cdots ; L_k ; not\, L_{k+1} ; \cdots ; not\, L_l \leftarrow L_{l+1}, \ldots, L_m, not\, L_{m+1}, \ldots, not\, L_n \quad (1)$$

where $0 \leq k \leq l \leq m \leq n$, each L_i is a literal, *not* is *negation as failure* (NAF), and ";" represents disjunction. A rule of the form (1) is an *(integrity) constraint* if $k = 0$, and is a *fact* if $k = l = m = n \geq 1$. A fact is *disjunctive* if $k > 1$. A fact is often denoted by omitting $\leftarrow$. A more general class of programs called *nested programs* [23] can be considered here, although any nested program is equivalent to some GEDP [23]. Similarly, any *propositional formula* can be used as a logic program [8]. Another important class of programs we can compare includes *logic programs with aggregates* [46, 27, 47]. Examples of subclasses of GEDPs are as follows. If any rule of the form (1) in a GEDP P satisfies $k = l$, P is called an *extended disjunctive program* (EDP) [10]. If any rule of the form (1) in an EDP P satisfies $k = l \leq 1$, P is called an *extended logic program* (ELP). An ELP P is a *normal logic program* (NLP) if every literal appearing in P is a positive literal. A program P is *NAF-free* if every rule of the form (1) in P never contains *not*, i.e., $k = l$ and $m = n$. A NAF-free program P is a *positive disjunctive program* (PDP) if every literal appearing in P is a positive literal. A PDP P is a *definite program* if $k = 1$ holds for any rule in P.

The definition of *answer sets* for each class of programs can be referred to the original papers mentioned above, so we will omit it here. The set of all answer sets of a program P is written as $A(P)$. A program P is *consistent* if $A(P)$ is consistent. It is known that $A(P)$ is irredundant for any EDP P, that is, every answer set of an EDP is *minimal* [10]. On the other hand, answer sets of GEDPs, nested programs and logic programs with aggregates are non-minimal in general. Notice also that we can consider any semantics other than the original answer set semantics for a logic program as long as it is defined as a closed composite set. Then, this setting allows comparison of EDPs under the *possible model semantics* [40] or the *supported model semantics* [14, 48], which has representation of non-minimal literal sets. Moreover, when PDPs P_1 and P_2 are compared, we can even consider their *classical models* $Mod(P_i)$ $(i = 1, 2)$ instead of their minimal models that are answer sets.

In the answer set semantics, a program containing variables is a shorthand of its ground instantiation. In the case of non-ground programs, however, we need

[6] *Stable models* are used to refer to answer sets of logic programs in which explicit negation ($\neg$) does not appear, and are sets of atoms instead of literals in this paper.

to assume that the relation $\supseteq$ is upwards well-founded on $\mathcal{P}(Lit)$ whenever the maximal elements are concerned (see Footnotes 3 and 4 in Section 2). Also the issue of decidability is of concern for non-ground programs [5, 22, 30]. To simplify the discussion, we assume that programs in this paper are finite propositional.

We denote by $\mathcal{LP}$ the class of all programs which can be constructed in the language. A subclass of programs is then denoted as $\mathcal{C} \subseteq \mathcal{LP}$. Examples of such subclasses $\mathcal{C}$ are the class of all programs $\mathcal{LP}$, the class of EDPs, the class of programs constructed with a subset of Lit, a set of programs, and a set of literal sets. The following correspondence notions have been proposed in the literature.

Definition 3.1 Let $P_1, P_2 \in \mathcal{LP}$ be programs, and $\mathcal{C} \subseteq \mathcal{LP}$.

(1) P_1 and P_2 are *(weakly) equivalent*, written $P_1 \equiv P_2$, if $A(P_1) = A(P_2)$ holds.
(2) P_1 and P_2 are *strongly equivalent*, written $P_1 \equiv_s P_2$, if $A(P_1 \cup R) = A(P_2 \cup R)$ holds for any program $R \in \mathcal{LP}$ [21].
(3) P_1 and P_2 are *uniformly equivalent*, written $P_1 \equiv_u P_2$, if $A(P_1 \cup U) = A(P_2 \cup U)$ holds for any literal set $U \in \mathcal{P}(Lit)$ [4].
(4) P_1 and P_2 are *(strongly) equivalent with respect to* $\mathcal{C}$, written $P_1 \equiv_{\mathcal{C}} P_2$, if $A(P_1 \cup R) = A(P_2 \cup R)$ holds for any $R \in \mathcal{C}$ [15].
(5) P_1 *ans-includes* P_2, written $P_1 \sqsupseteq P_2$, if $A(P_1) \supseteq A(P_2)$ holds [7].
(6) P_1 *strongly ans-includes* P_2, written $P_1 \sqsupseteq_s P_2$, if $A(P_1 \cup R) \supseteq A(P_2 \cup R)$ holds for any $R \in \mathcal{LP}$ [7]. P_1 *(strongly) ans-includes* P_2 *with respect to* $\mathcal{C}$, written $P_1 \sqsupseteq_{\mathcal{C}} P_2$, if $A(P_1 \cup R) \supseteq A(P_2 \cup R)$ holds for any $R \in \mathcal{C}$.

Relativized (strong) equivalence in Definition 3.1 (4) is the most general notion of equivalence in the sense that the other notions can be regarded as its special cases with particular contexts $\mathcal{C}$ [15].[7] That is, (1) weak, (2) strong and (3) uniform equivalence are the cases of $\mathcal{C} = \{\emptyset\}$, $\mathcal{C} = \mathcal{LP}$, and $\mathcal{C} = \mathcal{P}(Lit)$, respectively. The notion of *relativized (strong) inclusion* is likewise, and in particular we can define that P_1 *uniformly ans-includes* P_2 [31] iff P_1 strongly ans-includes P_2 with respect to $\mathcal{P}(Lit)$. We can further define the notions of *relativized uniform equivalence* [15, 49] and *relativized uniform inclusion* [31] as special cases of (4) relativized equivalence and (6) relativized inclusion, respectively, such that $\mathcal{C} = \mathcal{P}(U)$ for some literal set $U \subseteq Lit$.

Assuming a poset $\langle \mathcal{P}(Lit), \supseteq \rangle$, the Smyth and Hoare orderings have been defined on $\mathcal{P}(\mathcal{P}(Lit))$ in Section 2. Then, a program P is associated with its answer sets $A(P)$ to define generality orderings over the programs in $\mathcal{LP}$.

Definition 3.2 Assume the poset $\langle \mathcal{P}(Lit), \supseteq \rangle$, and let $P_1, P_2 \in \mathcal{LP}$. P_1 is *more (or equally)* $\sharp$*-general than* P_2, written $P_1 \models^{\sharp} P_2$, if $A(P_1) \models^{\sharp} A(P_2)$. P_1 is *more (or equally)* $\flat$*-general than* P_2, written $P_1 \models^{\flat} P_2$, if $A(P_1) \models^{\flat} A(P_2)$.

[7] This representation with a context $\mathcal{C}$ in Definition 3.1 (4) is similar to that in [48]. This is a slightly more general definition than that in [15], which can be represented by setting $\mathcal{C} = \mathcal{P}(R)$ for some rule set $R \in \mathcal{LP}$. Likewise, this is more general than the definition in [49, 50], which can be expressed by setting $\mathcal{C}$ as the class of programs constructed with some literal set $U \subseteq Lit$ [49] or as the class of programs that use possibly different alphabets for the heads and bodies of rules [50].

In Definition 3.2, because $A(P_i)$ is closed for each program P_i, it holds that, $P_1 \models^{\sharp/\flat} P_2$ iff $A(P_1) \succeq^{\sharp/\flat} A(P_2)$. Note that Definition 3.2 is essentially the same as [17, Definition 2.2], but now we allow comparison of programs having non-minimal answer sets, whereas [17] only considers EDPs.

Theorem 3.1 *Let $P_1, P_2 \in \mathcal{LP}$. If $P_1 \sqsupseteq P_2$, then $P_2 \models^{\sharp} P_1$ and $P_1 \models^{\flat} P_2$.*

Theorem 3.1 follows from Proposition 2.4, and shows that the inclusion relation [7] can be interpreted in terms of generality relations.

An equivalence class under the $\models^{\sharp/\flat}$ relation can be characterized by the next proposition, which follows from Theorem 2.2.

Theorem 3.2 *Let $P_1, P_2 \in \mathcal{LP}$.*

(1) *$P_1 \models^{\sharp} P_2$ and $P_2 \models^{\sharp} P_1$ iff $min(A(P_1)) = min(A(P_2))$.*
(2) *$P_1 \models^{\flat} P_2$ and $P_2 \models^{\flat} P_1$ iff $max(A(P_1)) = max(A(P_2))$.*

In the case of EDPs, as shown in [17], each equivalence class precisely characterizes the set of weakly equivalent EDPs.

Corollary 3.3 *Let P_1 and P_2 be EDPs. Then, the following three are equivalent:* (1) $P_1 \models^{\sharp} P_2 \models^{\sharp} P_1$; (2) $P_1 \models^{\flat} P_2 \models^{\flat} P_1$; (3) $P_1 \equiv P_2$.

Proof. This has been proved in [17], but can be easily proved as follows. For any EDP P, $A(P)$ is irredundant. Then the result follows from Corollary 2.3. □

Example 3.1 Consider the following programs:

$$
\begin{aligned}
P_1 &= \{\, p \leftarrow \mathit{not}\, q \,\},\\
P_2 &= \{\, p \leftarrow \mathit{not}\, q, \quad q \leftarrow \mathit{not}\, p \,\},\\
P_3 &= \{\, p;\ q \leftarrow \,\},\\
P_4 &= \{\, p;\ q \leftarrow, \quad p \leftarrow q, \quad q \leftarrow p \,\},\\
P_5 &= \{\, p;\ \mathit{not}\, p \leftarrow, \quad q;\ \mathit{not}\, q \leftarrow, \quad \leftarrow \mathit{not}\, p, \mathit{not}\, q \,\}.
\end{aligned}
$$

Then, $P_4 \models^{\sharp} P_1 \models^{\sharp} P_2$ and $P_4 \models^{\flat} P_2 \models^{\flat} P_1$ hold (see Example 2.1). $P_2 \equiv P_3$, and thus $P_2 \models^{\sharp} P_3 \models^{\sharp} P_2$ and $P_2 \models^{\flat} P_3 \models^{\flat} P_2$.

For P_5, $A(P_5) = \{\{p\}, \{q\}, \{p, q\}\}$ is not irredundant, where $\{p, q\}$ is non-minimal. Then, $P_1 \models^{\sharp} P_5 \models^{\sharp} P_2 \models^{\sharp} P_5$ and $P_5 \models^{\flat} P_4 \models^{\flat} P_5 \models^{\flat} P_2$ hold.

Minimal upper bounds (mubs) and *maximal lower bounds* (mlbs) have been important in the theory of generalization and in ILP [34, 28], and can be applied to coordination [44], composition [42] and consensus [43] of multiple agents [17]. We can show that both a minimal upper bound and a maximal lower bound of any pair of programs exist with respect to ♯- and ♭-generality orderings, but omit the details here.

As a result, we can induce the poset $\langle \mathbf{LP}^{\sharp/\flat}, \geqq^{\sharp/\flat} \rangle$, where $\mathbf{LP}^{\sharp/\flat}$ is the equivalence classes from $\langle \mathcal{LP}, \models^{\sharp/\flat} \rangle$ and $\geqq^{\sharp/\flat}$ is a partial order satisfying that

$[P_1] \geqq^{\sharp/\flat} [P_2]$ if $P_1 \models^{\sharp/\flat} P_2$ for any $P_1, P_2 \in \mathcal{LP}$ and $[P_1], [P_2] \in \mathbf{LP}^{\sharp/\flat}$. Then, this poset constitutes a complete lattice such that (1) a program P is in the top (resp. bottom) element of $\langle \mathbf{LP}^{\sharp}, \geqq^{\sharp} \rangle$ iff $A(P) = \emptyset$ (resp. $A(P) = \{\emptyset\}$), and that (2) a program P is in the top (resp. bottom) element of $\langle \mathbf{LP}^{\flat}, \geqq^{\flat} \rangle$ iff $A(P) = \{Lit\}$ (resp. $A(P) = \emptyset$).

When a logic program has multiple answer sets, the $\sharp$- and $\flat$-generality relations can be connected with skeptical and credulous entailment, respectively.

Definition 3.3 Let $P \in \mathcal{LP}$ be a program, and ψ a set of literals, which is interpreted as the conjunction of literals in ψ.[8] Then, ψ is a *skeptical consequence* of P if $\psi \subseteq S$ for every $S \in A(P)$. ψ is a *credulous consequence* of P if $\psi \subseteq S$ for some $S \in A(P)$. The sets of skeptical and credulous consequences of P are denoted as $Skp(P)$ and $Crd(P)$, respectively.

Proposition 3.4 *Let $P \in \mathcal{LP}$. If P is consistent, $Skp(P) = \bigcap_{S \in A(P)} \mathcal{P}(S)$ and $Crd(P) = \bigcup_{S \in A(P)} \mathcal{P}(S)$. If $A(P) = \emptyset$, $Skp(P) = \mathcal{P}(Lit)$ and $Crd(P) = \emptyset$. If $A(P) = \{Lit\}$, $Skp(P) = Crd(P) = \mathcal{P}(Lit)$.*

Theorem 3.5 *Let $P_1, P_2 \in \mathcal{LP}$.*

(1) *If $P_1 \models^{\sharp} P_2$ then $Skp(P_1) \supseteq Skp(P_2)$.*
(2) *$P_1 \models^{\flat} P_2$ iff $Crd(P_1) \supseteq Crd(P_2)$.*

Proof. (1) Assume that $P_1 \models^{\sharp} P_2$. If P_1 is inconsistent, then $Skp(P_1) = \mathcal{P}(Lit)$ and thus $Skp(P_1) \supseteq Skp(P_2)$. Suppose that P_1 is consistent and $\psi \in Skp(P_2)$. Then, $\psi \subseteq T$ for every $T \in A(P_2)$. By $P_1 \models^{\sharp} P_2$, for any $S \in A(P_1)$, there is $T' \in A(P_2)$ such that $S \supseteq T'$. Since $\psi \subseteq T'$, $\psi \subseteq S$ too. That is, $\psi \in Skp(P_1)$, and thus $Skp(P_1) \supseteq Skp(P_2)$.

(2) The only-if part can be proved in a similar way as (1). To prove the if part, suppose that $P_1 \not\models^{\flat} P_2$. Then, $A(P_1) \neq \{Lit\}$ and $A(P_2) \neq \emptyset$. Then, there is a set $T \in A(P_2)$ such that $S \not\supseteq T$ for any $S \in A(P_1)$. That is, for each $S \in A(P_1)$, we can pick a literal $\varphi_S \in (T \setminus S)$. Then, $\psi = \{\varphi_S \mid S \in A(P_1)\} \subseteq T$, whereas $\psi \not\subseteq S$ for any $S \in A(P_1)$. Hence, $\psi \in Crd(P_2)$ and $\psi \notin Crd(P_1)$, and thus $Crd(P_1) \not\supseteq Crd(P_2)$. □

By Theorem 3.5, the more $\sharp$-general (resp. $\flat$-general) a program is, the more it entails skeptically (resp. credulously). Hence, the Smyth and Hoare orderings over programs reflect the amount of information by skeptical and credulous entailment, respectively. Moreover, $\flat$-generality precisely reflects informativeness of credulous entailment. On the other hand, the converse of Theorem 3.5 (1) does not hold for $\sharp$-generality.[9] For example, for $A(P_1) = \{\{p\}\}$ and $A(P_2) = \{\{q\}, \{r\}\}$, $Skp(P_1) = \{\emptyset, \{p\}\}$ and $Skp(P_2) = \{\emptyset\}$ hold, and thus $Skp(P_1) \supset Skp(P_2)$ but $P_1 \not\models^{\sharp} P_2$.

[8] This definition extends the previous one in [17, Definition 4.1], in which a consequence was defined for each single literal instead of a conjunction of literals.

[9] The converse of Theorem 3.5 (1) would also hold if we could include *disjunctive facts* along with conjunctions of literals in the consequences. For example, for $A(P_2) = \{\{q\}, \{r\}\}$, the extended set of skeptical consequences of P_2 contains $q; r$.

Example 3.2 In Example 3.1, $A(P_3) = \{\{p\}, \{q\}\}$ and $A(P_4) = \{\{p, q\}\}$, and thus $P_4 \models^\sharp P_3$ and $P_4 \models^\flat P_3$. By Definition 3.3, $Skp(P_3) = \emptyset$, $Crd(P_3) = \{\emptyset, \{p\}, \{q\}\}$, and $Skp(P_4) = Crd(P_4) = \mathcal{P}(\{p, q\})$. Correspondingly, $Skp(P_4) \supset Skp(P_3)$ and $Crd(P_4) \supset Crd(P_3)$, which verify Theorem 3.5. Note here that $Crd(P_3) \neq Crd(P_4)$, because $\{p, q\}$ is only included in the latter. Note also that $Crd(P_4) = Crd(P_5)$, and correspondingly $P_5 \models^\flat P_4 \models^\flat P_5$.

4 Strong, Uniform and Relativized Generality

A possible issue in generality by Definition 3.2 is that generality of programs is determined solely by their answer sets. However, since the set of answer sets of a program does not monotonically decrease as the program grows, we often want to know how the generality relation between programs changes when some rules are added to those programs. In this section, we examine such context-dependent versions of generality relations. Then, *robustness* of generality between programs can be discussed by the property that one program brings more information than another whatever any set of rules in a focused context are added to those programs.

Another concern on generality is the question of whether or not a generality relation under consideration agrees with the property of classical generality relation defined in ILP for first-order logic. In first-order logic, a theory T_1 is defined to be *more (or equally) general* than T_2 if every formula derived from T_2 is derived from T_1, i.e., $T_1 \models T_2$ [34, 29]. Hence, classical generality is defined as the first-order entailment. Although the syntax of logic programming is different from first-order logic, it has been a convention that the class of PDPs and definite programs can also be regarded as a set of first-order formulas. We say that a generality relation $\succeq$ satisfies the *classical inductive generality* if it holds that $P_1 \succeq P_2$ iff $P_1 \models P_2$ for all PDPs P_1 and P_2. With this regard, the generality relation $\models^{\sharp/\flat}$ cannot make programs with the same answer sets different, hence does not satisfy the classical inductive generality. This property is often inconvenient from the viewpoint of program development.

Example 4.1 Suppose the three programs containing single rules, $P_1 = \{r_1\}$, $P_2 = \{r_2\}$ and $P_3 = \{r_3\}$, where

$$
\begin{aligned}
r_1 &: \ p \leftarrow q, \\
r_2 &: \ p \leftarrow q, r, \\
r_3 &: \ p \leftarrow q, \mathit{not}\, r.
\end{aligned}
$$

Because the answer sets of all programs are $\{\emptyset\}$, they are weakly equivalent, i.e., $P_1 \equiv P_2 \equiv P_3$. In ILP, however, P_1 is regarded as more general than P_2 because $P_1 \models P_2$ but $P_2 \not\models P_1$. In this case, under the classical models, $Mod(P_1) \models^\sharp Mod(P_2)$ and $Mod(P_2) \not\models^\sharp Mod(P_1)$ hold by $Mod(P_2) \supsetneq Mod(P_1)$ and Proposition 2.4. Similarly, P_1 is considered more general than P_3 in ILP. In program development, r_1 is often called a *generalization* of both r_2 and r_3, and conversely r_2 and r_3 are called *specializations* of r_1.

Example 4.1 indicates that, even in definite programs (P_1 and P_2), the generality relation $\models^{\sharp/\flat}$ does not satisfy the classical inductive generality as long as only the answer sets (i.e., minimal models) are concerned. This is because information contents brought by the collection of answer sets represents only the "solutions" of the current program, and "predictive (or deductive) power" of the program does not appear in the resultant answer sets but is implicit in the program itself. In this respect, too, we need to consider more context-sensitive notions of generality.

The notion of *strong generality* has already been introduced briefly in [17] as a counterpart of *strong equivalence* [21]. In this section, we firstly recall *strong generality* and newly consider *uniform* and *relativized* variants of generality relations in answer set programming. These three notions of generality are all context-sensitive variants of generality, but the strong and uniform versions can be regarded as special cases of relativized generality.

Definition 4.1 Let $P_1, P_2 \in \mathcal{LP}$ be programs, and $\mathcal{C} \subseteq \mathcal{LP}$.

(1) P_1 is *strongly more $\sharp$-general than* P_2, written $P_1 \trianglerighteq^{\sharp}_{s} P_2$, if $P_1 \cup R \models^{\sharp} P_2 \cup R$ for any program $R \in \mathcal{LP}$. P_1 is *strongly more $\flat$-general than* P_2, written $P_1 \trianglerighteq^{\flat}_{s} P_2$, if $P_1 \cup R \models^{\flat} P_2 \cup R$ for any program $R \in \mathcal{LP}$ [17].
(2) P_1 is *uniformly more $\sharp$-general than* P_2, written $P_1 \trianglerighteq^{\sharp}_{u} P_2$, if $P_1 \cup U \models^{\sharp} P_2 \cup U$ for any literal set $U \in \mathcal{P}(Lit)$. P_1 is *uniformly more $\flat$-general than* P_2, written $P_1 \trianglerighteq^{\flat}_{u} P_2$, if $P_1 \cup U \models^{\flat} P_2 \cup U$ for any literal set $U \in \mathcal{P}(Lit)$.
(3) P_1 is *(strongly) more $\sharp$-general than P_2 with respect to* $\mathcal{C}$, written $P_1 \trianglerighteq^{\sharp}_{\mathcal{C}} P_2$, if $P_1 \cup R \models^{\sharp} P_2 \cup R$ for any $R \in \mathcal{C}$. P_1 is *(strongly) more $\flat$-general than P_2 with respect to* $\mathcal{C}$, written $P_1 \trianglerighteq^{\flat}_{\mathcal{C}} P_2$, if $P_1 \cup R \models^{\flat} P_2 \cup R$ for any $R \in \mathcal{C}$.

The notions of *strong $\sharp/\flat$-generality* and *uniform $\sharp/\flat$-generality* are represented in Definition 4.1 (1) and (2), which can be expressed as special cases of Definition 4.1 (3) such that $\mathcal{C} = \mathcal{LP}$ and $\mathcal{C} = \mathcal{P}(Lit)$, respectively. Similarly, we can define the notion of *relativized uniform $\sharp/\flat$-generality* as a special case of Definition 4.1 (3) such that $\mathcal{C} = \mathcal{P}(U)$ for some $U \subseteq Lit$. In a special case, the context can be given as a set consisting of a single program as $\mathcal{C} = \{R\}$ for some $R \in \mathcal{LP}$. This corresponds to the concept of *relative generalization* in ILP [34], in which P_1 is called *more general than (or equal to) P_2 relative to background knowledge R*. Other relations among these definitions hold as follows.

Proposition 4.1 (1) *$P_1 \trianglerighteq^{\sharp/\flat}_{s} P_2$ implies $P_1 \trianglerighteq^{\sharp/\flat}_{u} P_2$.*
(2) *$P_1 \trianglerighteq^{\sharp/\flat}_{u} P_2$ implies $P_1 \models^{\sharp/\flat} P_2$, and hence $P_1 \trianglerighteq^{\sharp/\flat}_{s} P_2$ implies $P_1 \models^{\sharp/\flat} P_2$.*
(3) *For any $\mathcal{C} \subseteq \mathcal{LP}$, $P_1 \trianglerighteq^{\sharp/\flat}_{s} P_2$ implies $P_1 \trianglerighteq^{\sharp/\flat}_{\mathcal{C}} P_2$.*
(4) *For any $\mathcal{C} \subseteq \mathcal{LP}$ such that $\emptyset \in \mathcal{C}$, $P_1 \trianglerighteq^{\sharp/\flat}_{\mathcal{C}} P_2$ implies $P_1 \models^{\sharp/\flat} P_2$.*

Note that the condition $\emptyset \in \mathcal{C}$ is indispensable in Proposition 4.1 (4). For example, when $P_1 = \{p \leftarrow q\}$ and $P_2 = \{p \leftarrow\ \}$ are compared in the context $\mathcal{C} = \{\{q \leftarrow\ \}\}$, we have $P_1 \trianglerighteq^{\sharp}_{\mathcal{C}} P_2$ but $P_1 \not\models^{\sharp} P_2$.

It has been shown in [17] that programs belonging to the same equivalence class induced by strong generality become strongly equivalent in the case of

EDPs. We here see that similar results also hold for uniform and relativized generality.

Proposition 4.2 *Let P_1 and P_2 be EDPs, and $\mathcal{C}$ a set of EDPs. Then, in each of the following, the statements* (i)*,* (ii) *and* (iii) *are equivalent.*

(1) [17] (i) $P_1 \trianglerighteq_s^\sharp P_2 \trianglerighteq_s^\sharp P_1$; (ii) $P_1 \trianglerighteq_s^\flat P_2 \trianglerighteq_s^\flat P_1$; (iii) $P_1 \equiv_s P_2$.
(2) (i) $P_1 \trianglerighteq_u^\sharp P_2 \trianglerighteq_u^\sharp P_1$; (ii) $P_1 \trianglerighteq_u^\flat P_2 \trianglerighteq_u^\flat P_1$; (iii) $P_1 \equiv_u P_2$.
(3) (i) $P_1 \trianglerighteq_\mathcal{C}^\sharp P_2 \trianglerighteq_\mathcal{C}^\sharp P_1$; (ii) $P_1 \trianglerighteq_\mathcal{C}^\flat P_2 \trianglerighteq_\mathcal{C}^\flat P_1$; (iii) $P_1 \equiv_\mathcal{C} P_2$.

Theorem 4.3 *Let $P_1, P_2 \in \mathcal{LP}$. Suppose that Lit is finite.*
(1) *If $P_1 \trianglerighteq_s^\sharp P_2$ then $P_2 \sqsupseteq P_1$.* (2) *If $P_1 \trianglerighteq_s^\flat P_2$ then $P_1 \sqsupseteq P_2$.*

Proof. We prove (1), but (2) can be proved in a similar way.

Assume that $P_2 \not\sqsupseteq P_1$. Then, $A(P_2) \not\supseteq A(P_1)$, and hence $D = A(P_1) \setminus A(P_2) \neq \emptyset$. If there is an answer set $S \in D$ ($\subseteq A(P_1)$) such that $S \subset T$ for any answer set $T \in A(P_2)$, then $P_1 \not\models^\sharp P_2$ and thus $P_1 \not\trianglerighteq_s^\sharp P_2$. Otherwise, $S \not\subset T$ for any $S \in D$ and any $T \in A(P_2)$, and moreover $S \not\subseteq T$ by the fact that $T \notin D$ and hence $S \neq T$. Then, $S \setminus T \neq \emptyset$. Let $C = \{ \leftarrow not\, L_1, \ldots, not\, L_n \mid L_i \in (S_i \setminus T), D = \{S_1, \ldots, S_n\}, T \in A(P_2) \}$. Each constraint in C is always satisfied by some $S_i \in A(P_1)$, but can never be satisfied by any $T \in A(P_2)$. Hence, $A(P_1 \cup C) \neq \emptyset$ and $A(P_2 \cup C) = \emptyset$. Therefore, $P_1 \cup C \not\models^\sharp P_2 \cup C$, and hence $P_1 \not\trianglerighteq_s^\sharp P_2$. □

Theorem 4.3 states that *strong ♯/♭-generality implies inclusion*. This rather unexpected result can be explained by tracking the proof as follows. Simply assume that both P_1 and P_2 have single answer sets, $A(P_1) = \{S\}$ and $A(P_2) = \{T\}$. If $S \supset T$, the relation $P_1 \models^\sharp P_2$ holds. However, adding to P_1 and P_2 the constraint $C = \{ \leftarrow not\, L\}$ for some $L \in S \setminus T$ makes $P_2 \cup C$ inconsistent, and hence $P_2 \cup C \models^\sharp P_1 \cup C$ holds. Therefore, for P_1 to be strongly more ♯-general than P_2, it is necessary that $S \setminus T = \emptyset$. From Theorem 4.3, we can derive the next stronger result:[10] *strong generality is equivalent to strong inclusion.*

Theorem 4.4 *Let $P_1, P_2 \in \mathcal{LP}$, and suppose that Lit is finite. Then, $P_1 \trianglerighteq_s^\sharp P_2$ iff $P_2 \trianglerighteq_s^\flat P_1$ iff $P_2 \sqsupseteq_s P_1$.*

Proof. We prove ($P_1 \trianglerighteq_s^\sharp P_2$ iff $P_2 \sqsupseteq_s P_1$), but ($P_1 \trianglerighteq_s^\flat P_2$ iff $P_1 \sqsupseteq_s P_2$) can be proved in a similar way. Suppose $P_1 \trianglerighteq_s^\sharp P_2$. By definition, $P_1 \cup R \models^\sharp P_2 \cup R$ for any $R \in \mathcal{LP}$. Then, $(P_1 \cup R_1) \cup R_2 \models^\sharp (P_2 \cup R_1) \cup R_2$ for any $R_1, R_2 \in \mathcal{LP}$. By definition, $P_1 \cup R_1 \trianglerighteq_s^\sharp P_2 \cup R_1$ for any $R_1 \in \mathcal{LP}$. By Theorem 4.3 (1), $A(P_2 \cup R_1) \supseteq A(P_1 \cup R_1)$ for any $R_1 \in \mathcal{LP}$. Hence, $P_2 \sqsupseteq_s P_1$. Conversely, suppose $P_2 \sqsupseteq_s P_1$. By definition, $A(P_2 \cup R) \supseteq A(P_1 \cup R)$ for any $R \in \mathcal{LP}$. By Theorem 3.1, $P_1 \cup R \models^\sharp P_2 \cup R$ for any $R \in \mathcal{LP}$. Hence, $P_1 \trianglerighteq_s^\sharp P_2$. □

Example 4.2 Consider P_1, P_2 and P_3 in Example 4.1. For $R_1 = \{ q \leftarrow not\, p \}$, $A(P_1 \cup R_1) = \emptyset$, while $A(P_2 \cup R_1) = \{\{q\}\}$. By contrast, for $R_2 = \{ q \leftarrow ,\ \ \leftarrow not\, p \}$,

[10] This property was pointed out to the authors by Jianmin Ji for EDPs.

$A(P_1 \cup R_2) = \{\{p, q\}\}$, while $A(P_2 \cup R_2) = \emptyset$. Hence, $P_1 \not\trianglerighteq_s^\sharp P_2$ and $P_2 \not\trianglerighteq_s^\sharp P_1$. Similarly, for $R_3 = \{\, r \leftarrow ,\ q \leftarrow not\, p\,\}$, $A(P_1 \cup R_3) = \emptyset$ and $A(P_3 \cup R_3) = \{\{q, r\}\}$. However, for $R_4 = \{\, q \leftarrow ,\ \ r \leftarrow ,\ \ \leftarrow not\, p\,\}$, $A(P_1 \cup R_4) = \{\{p, q, r\}\}$ and $A(P_3 \cup R_4) = \emptyset$. Hence, $P_1 \not\trianglerighteq_s^\sharp P_3$ and $P_3 \not\trianglerighteq_s^\sharp P_1$.

Example 4.2 indicates that strong generality does not satisfy the classical inductive generality even for definite programs.

On the other hand, we see that both $P_1 \trianglerighteq_u^\sharp P_2$ and $P_1 \trianglerighteq_u^\sharp P_3$ hold. Uniform generality thus satisfies the classical inductive generality for Example 4.1. Indeed, the next result states that uniform generality collapses to classical entailment in the case of PDPs. Related results are stated in [26, 4] such that uniform equivalence coincides with classical equivalence for PDPs.

Proposition 4.5 *Let P_1 and P_2 be PDPs. Then, $P_1 \trianglerighteq_u^\sharp P_2$ iff $P_1 \models P_2$, that is, P_1 classically entails P_2.*

Proof. It is easy to see that $P_1 \models P_2$ implies $P_1 \trianglerighteq_u^\sharp P_2$. Conversely, suppose that $P_1 \not\models P_2$. Then, there is a set M of ground atoms such that $M \in Mod(P_1)$ and $M \notin Mod(P_2)$. Let $M' = M \cup \{\, \neg A \in Lit \mid A \notin M\}$. Since M is not a model of P_2, $P_2 \cup M'$ is inconsistent. By Herbrand's theorem, there is a finite subset $U \subseteq M'$ of ground literals such that $P_2 \cup U$ is inconsistent. Since M is a model of P_1, $P_1 \cup U$ is consistent. Then, there is a consistent answer set of $P_1 \cup U$, but Lit is the unique inconsistent answer set of $P_2 \cup U$. Obviously, $P_1 \cup U \not\models^\sharp P_2 \cup U$. Hence, $P_1 \not\trianglerighteq_u^\sharp P_2$. □

Proposition 4.6 *Let P_1 and P_2 be PDPs without integrity constraints. Then, $P_1 \trianglerighteq_u^\flat P_2$ iff $P_1 \models P_2$.*

Proof. The property (if $P_1 \trianglerighteq_u^\flat P_2$ then $P_1 \models P_2$) can be proved in the same way as Proposition 4.5. To show that (if $P_1 \models P_2$ then $P_1 \trianglerighteq_u^\flat P_2$), we need the condition that P_1 does not contain integrity constraints. When P_1 has no constraint, $P_1 \cup U$ always has an answer set for any literal set $U \in \mathcal{P}(Lit)$. Then, $P_1 \cup U \models^\flat P_2 \cup U$. □

Example 4.3 Consider P_1, P_2 and P_3 in Example 4.1. We see that both $P_1 \trianglerighteq_{\mathcal{C}}^\sharp P_2$ and $P_1 \trianglerighteq_{\mathcal{C}}^\sharp P_3$ hold for $\mathcal{C} = \mathcal{P}(R)$, where $R = \{p,\ q,\ r,\ p; q,\ p; r,\ q; r,\ p; q; r\}$.

Example 4.3 strengthens the classical inductive property for uniform generality to that for relativized generality with the context $\mathcal{C}$ including disjunctive facts. Moreover, relativized generality also holds when the context $\mathcal{C}$ is the class of PDPs.

Proposition 4.7 *Let $\mathcal{C}$ be the class of PDPs. Then, for any two PDPs $P_1, P_2 \in \mathcal{C}$, $P_1 \trianglerighteq_{\mathcal{C}}^\sharp P_2$ iff $P_1 \models P_2$.*

Proof. $P_1 \models P_2$ implies $P_1 \cup R \models^\sharp P_2 \cup R$ for any $R \in \mathcal{C}$. Hence, $P_1 \models P_2$ implies $P_1 \trianglerighteq_{\mathcal{C}}^\sharp P_2$. The converse property ($P_1 \not\models P_2$ implies $P_1 \not\trianglerighteq_{\mathcal{C}}^\sharp P_2$) can be proved in the same way as Proposition 4.5. □

Proposition 4.8 *Let $\mathcal{C}$ be the class of PDPs having no integrity constraint. Then, for any two PDPs $P_1, P_2 \in \mathcal{C}$, $P_1 \trianglerighteq^{\flat}_{\mathcal{C}} P_2$ iff $P_1 \models P_2$.*

Proof. Let R be any PDP in $\mathcal{C}$. Since P_1, P_2 and R are PDPs containing no constraints, $P_1 \cup R$ has always an answer set, and thus $P_1 \models P_2$ implies $P_1 \cup R \models^{\flat} P_2 \cup R$. Hence, $P_1 \models P_2$ implies $P_2 \trianglerighteq^{\flat}_{\mathcal{C}} P_1$. The converse direction can be proved in the same way as Proposition 4.6. □

In both Propositions 4.6 and 4.8, the condition that P_1 does not contain an integrity constraint is necessary to establish the relations that $P_1 \models P_2$ implies $P_1 \trianglerighteq^{\flat}_{u} P_2$ and $P_1 \trianglerighteq^{\flat}_{\mathcal{C}} P_2$. For example, when $P_1 = \{ \leftarrow a \}$ and $P_2 = \emptyset$, $P_1 \models P_2$ holds. However, for $U = \{ a \leftarrow \}$, $A(P_1 \cup U) = \emptyset$ and $A(P_2 \cup U) = \{\{a\}\}$, and hence $P_1 \not\trianglerighteq^{\flat}_{u} P_2$ and $P_1 \not\trianglerighteq^{\flat}_{\mathcal{C}} P_2$.

5 Abductive Generality

In this section, we show that relativized generality in the previous section can be well applied to generality relations for *abductive logic programming* (ALP) [20]. Then, such generality relations could be applied to give logical foundations of coordination between multiple *abductive agents* by extending results of [44, 42, 43] to combination of abductive programs.

A semantics for ALP is given by extending answer sets of the background program with addition of abducibles. Such an extended answer set is called a *generalized stable model* [20] or a *belief set* [16].

Definition 5.1. An *abductive (logic) program* is a pair $\langle P, \Gamma \rangle$, where $P \in \mathcal{LP}$ is a logic program and $\Gamma \subseteq \mathit{Lit}$ is a set of literals called *abducibles*.[11] Put $\mathbf{\Gamma} = \mathcal{P}(\Gamma)$.

Let $A = \langle P, \Gamma \rangle$ be an abductive program, and $E \in \mathbf{\Gamma}$, i.e., $E \subseteq \Gamma$. A *belief set* of A (*with respect to* E) is a consistent answer set of the logic program $P \cup E$. The set of all belief sets of A is denoted as $B(A)$. A set $S \in B(A)$ is often denoted as S_E when S is a belief set with respect to E.

An *observation* G is defined as a set of ground literals, which is interpreted as the conjunction of its literals. A set $E \in \mathbf{\Gamma}$ is a *brave* (resp. *cautious*) *explanation* of G *in* A if G is true in some (resp. every) belief set of A with respect to E.[12] When G has a brave (resp. cautious) explanation in A, G is *bravely* (resp. *cautiously*) *explainable* in A.

Note that $B(A)$ is a composite set not containing Lit. Inoue and Sakama [19] introduce two measures for comparing (brave) explanation power of abductive

[11] The abducibles can be defined as $\Gamma \in \mathcal{LP}$ by allowing rules, but such an extended form of abductive program $\langle P, \Gamma \rangle$ can be reduced to an abductive program in Definition 5.1 by moving each rule in Γ to P with a new abuducible literal added to the body so that adding the new abducible enables the rule in abduction [12].

[12] Brave and cautious explanations are also called *credulous* and *skeptical* explanations, respectively. The properties of cautious explanations have not been studied in [19] and are newly investigated in this paper.

programs. One is aimed at comparing *explainability* for observations in different theories, while the other is aimed at comparing *explanation contents* for observations.

Definition 5.2. A_1 is *more (or equally) bravely* (resp. *cautiously*) *explainable than* A_2, written $A_1 \geqslant^b A_2$, (resp. $A_1 \geqslant^c A_2$), if every observation bravely (resp. cautiously) explainable in A_2 is also bravely (resp. cautiously) explainable in A_1.

On the other hand, A_1 is *more (or equally) bravely* (resp. *cautiously*) *explanatory than* A_2, written $A_1 \ni^b A_2$ (resp. $A_1 \ni^c A_2$), if, for any observation G, every brave (resp. cautious) explanation of G in A_2 is also a brave (resp. cautious) explanation of G in A_1.

Note that $A_1 \ni^b A_2$ implies $A_1 \geqslant^b A_2$ and that $A_1 \ni^c A_2$ implies $A_1 \geqslant^c A_2$. To simplify the problem, we hereafter assume that $A_1 = \langle P_1, \Gamma \rangle$ and $A_2 = \langle P_2, \Gamma \rangle$ are abductive programs with the same abducibles Γ.

Example 5.1 Let $A_1 = \langle P_1, \Gamma \rangle$ and $A_2 = \langle P_2, \Gamma \rangle$ be abductive programs, where $P_1 = \{\, p \leftarrow a,\ a \leftarrow b \,\}$, $P_2 = \{\, p \leftarrow a,\ p \leftarrow b \,\}$, and $\Gamma = \{a, b\}$. Then, $A_1 \geqslant^b A_2$ and $A_2 \geqslant^b A_1$, while $A_1 \ni^b A_2$ but $A_2 \not\ni^b A_1$. In fact, $\{b\}$ is an explanation of a in A_1, but is not in A_2. So with the relations $\geqslant^c$ and $\ni^c$.

The relationships between the notions of abductive generality and (relativized) generality relations for logic programs can be established as follows. Note that these relationships are not obvious from the definitions.

Theorem 5.1 *$A_1 \geqslant^b A_2$ iff $B(A_1) \models^\flat B(A_2)$.*

Proof. This result follows from the property proved in [19] that, $A_1 \geqslant^b A_2$ holds iff for any belief set S_2 of A_2, there is a belief set S_1 of A_1 such that $S_1 \supseteq S_1$. □

An abductive program $A = \langle P, \Gamma \rangle$ can be translated to a logic program as follows [14]. Let $Cons = \{\, \leftarrow L, \neg L \mid L \in Lit \,\}$ [21], and put $P^+ = P \cup Cons$. Next, let $Abd(\Gamma) = \{\, L; not\ L \leftarrow \ \mid\ L \in \Gamma \,\}$, and put $P^+_\Gamma = P^+ \cup Abd(\Gamma)$. Then, $B(A) = A(P^+_\Gamma)$ holds [14]. With this result we get the next.

Corollary 5.2 *Let $A_1 = \langle P_1, \Gamma \rangle$ and $A_2 = \langle P_2, \Gamma \rangle$ be abductive programs. Then, $A_1 \geqslant^b A_2$ iff ${P_1}^+_\Gamma \models^\flat {P_2}^+_\Gamma$.*

Theorem 5.3 *Let $A_1 = \langle P_1, \Gamma \rangle$ and $A_2 = \langle P_2, \Gamma \rangle$ be abductive programs. Then, $A_1 \ni^b A_2$ iff ${P_1}^+ \trianglerighteq^\flat_{\mathbf{\Gamma}} {P_2}^+$.*

Proof. $A_1 \ni^b A_2$ iff for any $E \in \mathbf{\Gamma}$ and any $S_E \in B(A_2)$, there is $T_E \in B(A_1)$ such that $T_E \supseteq S_E$ [19] iff for any $E \in \mathbf{\Gamma}$, for any $S \in A(P_2^+ \cup E)$ there is $T \in A(P_1^+ \cup E)$ such that $T \supseteq S$. Hence, the result follows. □

Corollary 5.2 and Theorem 5.3 give us a better intuition on how explainable and explanatory generality are different in brave abduction: the former is characterized by $\flat$-generality, while the latter by relativized $\flat$-generality.

By contrast, explanatory (and explainable) generality in cautious abduction can be characterized by relativized strong $\sharp$-generality, but only as a sufficient condition.

Theorem 5.4 *Let* $A_1 = \langle P_1, \Gamma \rangle$ *and* $A_2 = \langle P_2, \Gamma \rangle$ *be abductive programs. If* ${P_1}^+ \trianglerighteq^\sharp_{\mathbf{\Gamma}} {P_2}^+$ *then* $A_1 \ni^c A_2$ *(and then* $A_1 \geqslant^c A_2$*).*

Proof. If ${P_1}^+ \trianglerighteq^\sharp_{\mathbf{\Gamma}} {P_2}^+$ then ${P_1}^+ \cup E \models^\sharp {P_2}^+ \cup E$ for any $E \in \mathbf{\Gamma}$. Then, $Skp({P_1}^+ \cup E) \supseteq Skp({P_2}^+ \cup E)$ for any $E \in \mathbf{\Gamma}$ by Theorem 3.5 (1). Hence, $A_1 \ni^c A_2$. □

Note that the converse of Theorem 5.4 does not hold as the converse of Theorem 3.5 (1) used in the proof does not hold.

Inoue and Sakama [16] study two equivalence relations in abduction called *explainable/explanatory equivalence*. Pearce *et al.* [33] characterize a part of these problems in the context of equilibrium logic. Since these abductive equivalence notions are defined in terms of brave explanations, they are related with brave abductive generality notions in [19]. Formally, two abductive programs A_1 and A_2 are *explainably equivalent* if, for any observation O,[13] O is explainable in A_1 iff O is explainable in A_2. On the other hand, A_1 and A_2 are *explanatorily equivalent* if, for any observation O, E is an explanation of O in A_1 iff E is an explanation of O in A_2.

Characterization of abductive equivalence has already been investigated in depth in [16, 19]. Using results in Sections 2 and 4, these relations can easily be derived as corollaries of Theorems 5.1 and 5.3 as follows.

Corollary 5.5 *Let* $A_1 = \langle P_1, \Gamma \rangle$ *and* $A_2 = \langle P_2, \Gamma \rangle$ *be abductive programs.*

(1) [19] A_1 *and* A_2 *are explainably equivalent iff* $max(B(A_1)) = max(B(A_2))$.
(2) [19] A_1 *and* A_2 *are explanatorily equivalent iff* ${P_1}^+ \trianglerighteq^\flat_{\mathbf{\Gamma}} {P_2}^+ \trianglerighteq^\flat_{\mathbf{\Gamma}} {P_1}^+$ *iff* $max(A(P_1^+ \cup E)) = max(A(P_2^+ \cup E))$ *for any* $E \in \mathbf{\Gamma}$.
(3) [16] *Suppose that both* P_1 *and* P_2 *are EDPs. Then,* A_1 *and* A_2 *are explanatorily equivalent iff* $P_1^+ \equiv_{\mathbf{\Gamma}} P_2^+$.

6 Discussion

The inclusion relation in answer set programming was firstly considered in Eiter *et al.* [7]. It is argued in [7] that, if every answer set of a program P is also an answer set of a program Q, i.e., $Q \sqsupseteq P$, then Q can be viewed as a *skeptically sound approximation* of P, meaning that $Skp(P) \supseteq Skp(Q)$. However, a program which has no inclusion relation with P can be a skeptically sound approximation of P as well. For example, suppose two programs $P = \{p \leftarrow\ ,\ q \leftarrow p\}$ and $Q = \{p \leftarrow\}$. Then, $A(P) = \{\{p, q\}\}$ and $A(Q) = \{\{p\}\}$, and hence Q is sound with respect to skeptical reasoning from P although the inclusion relation $\sqsupseteq$ does not hold between P and Q. Using generality, we can conclude that $P \models^\sharp Q$.

Section 2 gives a fairly general framework for comparing semantic structures called *composite sets*. This notion can be further extended by allowing any theory as an element of a composite set. For example, if a composite set is defined as

[13] This definition of explainable equivalence for ALP is not exactly the same as that in [16, Definition 4.3]. In [16] an observation is a single ground literal, while we allow a conjunction of ground literals as an observation.

a set of closed first-order theories, it can be applied to represent the semantic structure of *default logic* [36]. In Example 4.1, we have seen weakly equivalent programs, P_1, P_2 and P_3. As long as each rule in a program is interpreted as a first-order formula, the models of P_1 include those of P_2, and hence satisfy the classical inductive generality. With this regard, generality relations over default theories, which allow first-order formulas along with default rules, are proposed in [18]. Then, P_1 is properly ordered to be more general than P_2 if these programs are regarded as default theories. Unfortunately, the approach of [18] cannot differentiate P_1 and P_3 because r_1 can be interpreted as either a propositional formula or a default but r_3 can only be regarded as a default. We have seen that they are properly ordered using uniform generality and relativized generality.

We have introduced the notion of composite sets and their orderings to compare answer set programs in this paper. Since any pre-order $\succcurlyeq$ can be considered in an underlying pre-ordered set, an application of ranking or preferring composite sets to *decision theory* would be interesting in AI and economics. In this context, the $\flat$-ordering has been used to extend a preference relation over a possibly infinite set X to its powerset $\mathcal{P}(X)$ [1]. The comparison framework proposed in this paper could be the basis for ordering answer set programs in such a decision theoretic context. For example, we can compare two *prioritized logic programs* [41] by setting the pre-order $\succcurlyeq$ as the preference relations between answer sets.

There are some other approaches to order logic programs in the literature. In [38], answer set programs are ordered according to multi-valued interpretations for programs. A promising approach is to order programs according to the implication relation with *HT-models* based on the logic of here-and-there [2, 52]. A related approach is to use *SE-models* instead of HT-models for defining entailment and consequence relations [51, 3]. As is stated in [52], any generality relation in [17] does not coincide with any of [52]. Considering relativized versions of generality in this paper, however, it is worth investigating precise relations between these approaches and ours. In fact, as far as consistent programs are compared, the ordering based on HT-models results in a similar ordering to our strong generality.

7 Conclusion

In this paper, we have extended the $\sharp$- and $\flat$-generality orderings of [17] in various ways. The original contributions in this paper are summarized as follows.

- Applicability of comparison under generality is extended from EDPs to any class of logic programs including nested programs and logic programs with aggregates. In comparison under $\sharp$- (resp. $\flat$-) generality, it is shown that the minimal (resp. maximal) answer sets determine the generality relation.
- Strong, uniform and relativized generality for logic programs are proposed as context-sensitive versions of generality, and several relations between them are investigated. For example, we have shown that:
  - Every generality relation is an instance of relativized generality.

  - Strong generality coincides with strong inclusion.
  - Uniform generality for PDPs collapses to classical entailment.
  - Relativized equivalent EDPs can be characterized as an equivalence class induced by relativized $\sharp$ or $\flat$-generality.
- The notions of explainable and explanatory generality in abductive logic programming are characterized using $\flat$-generality and relativized $\flat$-generality, which provide us a better intuition on how they are different.

As for the computational complexity of the proposed framework, we can apply several previously known results to establish complexity results for our framework. Since our comparison framework is general enough to include many known equivalence and inclusion relations, hardness results can be easily obtained from complexity results for such subproblems, e.g., [4, 7, 31, 19]. On the other hand, those necessary and sufficient conditions such as Theorem 4.4, Propositions 4.5 and 4.7, Corollary 5.2 and Theorem 5.3 can be used to establish completeness results. Roughly speaking, the complexity classes range from coNP to Π_3^P in the polynomial hierarchy, and more precise characterizations will be reported elsewhere.

Other important future work includes incorporation of the notion of *projection* used in the framework of [7, 31, 35] into our generality framework and more investigation of generality in the non-ground case like studies for equivalence in [25, 5, 22, 30].

References

1. M. A. Ballester, J. R. De Miguel, Extending an order to the power set: the leximax criterion, *Social Choice and Welfare* 21 (2003) 63–71.
2. P. Cabalar, D. Pearce, A. Valverde, Minimal logic programs, in: *Proceedings ICLP '07*, LNCS 4670, Springer, 2007, pp. 104–118.
3. J. Delgrande, T. Schaub, H. Tompits, S. Woltran, Merging logic programs under answer set semantics, in: *Proceedings ICLP '09*, LNCS 5649, Springer, 2009, pp. 160–174.
4. T. Eiter, M. Fink, Uniform equivalence of logic programs under the stable model semantics, in: *Proceedings ICLP '03*, LNCS 2916, Springer, 2003, pp. 224–238.
5. T. Eiter, M. Fink, H. Tompits, S. Woltran, Strong and uniform equivalence in answer-set programming: characterizations and complexity results for the non-ground case, in: *Proceedings AAAI-05*, 2005, pp. 695–700.
6. T. Eiter, T. Lukasiewicz, R. Schindlauer, H. Tompits, Combining answer set programming with description logics for the semantic web, in: *Proceedings KR '04*, AAAI Press, 2004, pp. 141–151.
7. T. Eiter, H. Tompits, S. Woltran, On solution correspondences in answer-set programming, in: *Proceedings IJCAI-05*, 2005, pp. 97–102.
8. P. Ferraris, Answer sets for propositional theories, in: *Proceedings 8th International Conference on Logic Programming and Nonmonotonic Reasoning*, LNAI 3662, Springer, 2005, pp. 119–131.
9. M. Gelfond, V. Lifschitz. The stable model semantics for logic programming, in: *Proceedings ICLP '88*, MIT Press, 1988, pp. 1070–1080.

10. M. Gelfond, V. Lifschitz, Classical negation in logic programs and disjunctive databases, *New Generation Computing* 9 (1991) 365–385.
11. C. A. Gunter, D. S. Scott, Semantic domains, in: J. van Leeuwen (Ed.), *Handbook of Theoretical Computer Science*, Vol. B, North-Holland, 1990, pp. 633–674.
12. K. Inoue, Hypothetical reasoning in logic programs, *J. Logic Programming* 18 (1994) 191–227.
13. K. Inoue, Y. Kudoh, Learning extended logic programs, in: *Proceedings IJCAI-97*, Morgan Kaufmann, 1997, pp. 176–181.
14. K. Inoue, C. Sakama, Negation as failure in the head, *J. Logic Programming* 35 (1998) 39–78.
15. K. Inoue, C. Sakama, Equivalence of logic programs under updates, in: *Proceedings JELIA '04*, LNAI 3229, Springer, 2004, pp. 174–186.
16. K. Inoue, C. Sakama, Equivalence in abductive logic, in: *Proceedings IJCAI-05*, 2005, pp. 472–477.
17. K. Inoue, C. Sakama, Generality relations in answer set programming, in: *Proceedings ICLP '06*, LNCS 4079, Springer, 2006, pp. 211–225.
18. K. Inoue, C. Sakama, Generality and equivalence relations in default logic, in: *Proceedings AAAI-07*, 2007, pp. 434–439.
19. K. Inoue, C. Sakama, Comparing abductive theories, in: *Proceedings 18th European Conference on Artificial Intelligence*, 2008, pp. 35–39.
20. A. Kakas, R. Kowalski, F. Toni, The role of abduction in logic programming, in: D. Gabbay, C. Hogger, J. Robinson (Eds.), *Handbook of Logic in Artificial Intelligence and Logic Programming, Vol. 5*, pp. 235–324, Oxford University Press, 1998.
21. V. Lifschitz, D. Pearce, A. Valverde, Strongly equivalent logic programs, *ACM Transactions on Computational Logic* 2 (2001) 526–541.
22. V. Lifschitz, D. Pearce, A. Valverde, A characterization of strong equivalence for logic programs with variables, in: *Proceedings 9th International Conference on Logic Programming and Nonmonotonic Reasoning*, LNAI 4483, Springer, 2007, pp. 188–200.
23. V. Lifschitz, L. R. Tang, H. Turner, Nested expressions in logic programs, *Annals of Mathematics and Artificial Intelligence* 25 (1999) 369–389.
24. V. Lifschitz, T. Y. C. Woo, Answer sets in general nonmonotonic reasoning (preliminary report), in: *Proceedings KR '92*, Morgan Kaufmann, 1992, pp. 603–614.
25. F. Lin, Reducing strong equivalence of logic programs to entailment in classical propositional logic, In: *Proceedings KR '02*, 2002, pp. 170–176.
26. M. J. Maher, Equivalence of logic programs, in: J. Minker (Ed.), *Foundations of Deductive Databases and Logic Programming*, 1988, pp. 627–658.
27. V. W. Marek, I. Niemelä, M. Truszczyński, Logic programs with monotone abstract constraint atoms, *Theory and Practice of Logic Programming* 8 (2007) 167–199.
28. S. Muggleton, L. de Raedt, Inductive logic programming: theory and methods, *J. Logic Programming* 19/20 (1994) 629–679.
29. S.-H. Nienhuys-Cheng, R. de Wolf, *Foundations of Inductive Logic Programming*, LNAI 1228, Springer, 1997.
30. J. Oetsch, H. Tompits, Program correspondence under the answer-set semantics: the non-ground case, in: *Proceedings ICLP '08*, LNCS 5366, Springer, 2008, pp. 591–605.
31. J. Oetsch, H. Tompits, S. Woltran, Facts do not cease to exist because they are ignored: relativised uniform equivalence with answer-set projection, in: *Proceedings AAAI-07*, 2007, pp. 458–464.
32. R. Otero, Induction of stable models, in: *Proceedings 11th International Conference on Inductive Logic Programming*, LNAI 2157, Springer, 2001, pp. 193–205.

33. D. Pearce, H. Tompits, S. Woltran, Relativised equivalence in equilibrium logic and its applications to prediction and explanation: preliminary report, in: *Proceedings LPNMR'07 Workshop on Correspondence and Equivalence for Nonmonotonic Theories*, 2007, pp. 37–48.
34. G. D. Plotkin, A note on inductive generalization, in: B. Meltzer, D. Michie (Eds.), *Machine Intelligence* 5, Edinburgh University Press, 1970, pp. 153–63.
35. J. Pührer and H. Tompits, Casting away disjunction and negation under a generalisation of strong equivalence with projection, in: *Proceedings 10th International Conference on Logic Programming and Nonmonotonic Reasoning*, LNAI 5753, Springer, 2009, pp. 264–276.
36. R. Reiter, A logic for default reasoning, *Artificial Intelligence* 13 (1980) 81–132.
37. Y. Sagiv, Optimizing datalog programs, in: J. Minker (Ed.), *Foundations of Deductive Databases and Logic Programming*, 1988, pp. 659–668.
38. C. Sakama, Ordering default theories and nonmonotonic logic programs, *Theoretical Computer Science* 338 (2005) 127–152.
39. C. Sakama, Induction from answer sets in nonmonotonic logic programs, *ACM Transactions on Computational Logic* 6 (2005) 203–221.
40. C. Sakama, K. Inoue, An alternative approach to the semantics of disjunctive logic programs and deductive databases, *J. Automated Reasoning* 13 (1994) 145–172.
41. C. Sakama, K. Inoue, Prioritized logic programming and its application to commonsense reasoning, *Artificial Intelligence* 123 (2000) 185–222.
42. C. Sakama, K. Inoue, Combining answer sets of nonmonotonic logic programs, in: *Post-proceedings 6th International Workshop on Computational Logic in Multi-Agent Systems*, LNAI 3900, Springer, 2006, pp. 320-339.
43. C. Sakama, K. Inoue, Constructing consensus logic programs, in: *Post-proceedings 16th International Conference on Logic-based Program Synthesis and Transformation*, LNCS 4407, Springer, 2007, pp. 26–42.
44. C. Sakama, K. Inoue, Coordination in answer set programming, *ACM Transactions on Computational Logic* 9(2) (2008) A9.
45. C. Sakama, K. Inoue, Brave induction: a logical framework for learning from incomplete information, *Machine Learning* 76(1) (2009) 3–35.
46. P. Simons, I. Niemelä, T. Soininen, Extending and implementing the stable model semantics, *Artificial Intelligence* 138 (2002) 181–234.
47. T. C. Son, E. Pontelli, P. H. Tu, Answer sets for logic programs with arbitrary abstract constraint atoms, *J. Artificial Intelligence Research* 29 (2007) 353–389.
48. M. Truszczyński, S. Woltran, Hyperequivalence of logic programs with respect to supported models, in: *Proceedings AAAI-08*, 2008, pp. 560–565.
49. S. Woltran, Characterizations for relativized notions of equivalence in answer set programming, in: *Proceedings JELIA '04*, LNAI 3229, Springer, 2004, pp. 161–173.
50. S. Woltran, A common view on strong, uniform, and other notions of equivalence in answer set programming, *Theory and Practice of Logic Programming* 8 (2008) 217–234.
51. K.-S. Wong, Sound and complete inference rules for SE-consequences, *Journal of Artificial Intelligence Research* 31 (2008) 205–216.
52. Y. Zhou, Y. Zhang, Rule calculus: semantics, axioms and applications, in: *Proceedings JELIA '08*, LNAI 5293, Springer, 2008, pp. 416–428.