Available online at www.sciencedirect.com
SCIENCE DIRECT®
Theoretical Computer Science 338 (2005) 127–152
Theoretical Computer Science
www.elsevier.com/locate/tcs

Ordering default theories and nonmonotonic logic programs

Chiaki Sakama*

Department of Computer and Communication Sciences, Wakayama University, Sakaedani, Wakayama 640 8510, Japan

Received 25 December 2003; received in revised form 22 September 2004; accepted 1 December 2004

Communicated by G. Levi

Abstract

First-order theories are ordered under logical entailment based on the amount of information derived from theories. In *default logic*, on the other hand, a theory contains default information as well as definite information. To order default theories, distinguishing different sorts of information is necessary to assess the information content of a default theory. For this purpose, we first introduce a multi-valued interpretation of default theories using a ten-valued *bilattice*. It distinguishes between definite and credulous/skeptical default information derived from a theory, and is used for ordering default theories based on their information contents. We then apply the technique to order *nonmonotonic logic programs* under the *answer set semantics*. The results of this paper provide a method for comparing default theories or nonmonotonic logic programs in a manner different from the conventional extension/model-based viewpoint. Moreover, they have important application to induction from nonmonotonic theories.

Keywords: Bilattice; Default logic; Logic programming; Multi-valued logic

1. Introduction

In knowledge representation based on logic, different theories are compared by the amount of information derived from them. In first-order logic, a theory T_1 has more

* Tel./fax: +81 73 457 8128.
E-mail address: sakama@sys.wakayama-u.ac.jp (C. Sakama).

0304-3975/$ - see front matter © 2004 Elsevier B.V. All rights reserved.
doi:10.1016/j.tcs.2004.12.003

information than another theory T_2 if T_1 has fewer models than T_2 [18]. The relation is represented using logical entailment as $T_1 \models T_2$ and $T_2 \not\models T_1$, i.e., every formula derived from T_2 is also derived from T_1 but not vice versa. In this case, a theory T_1 is said *stronger* than T_2. For instance, the theory

$$T_1: \quad bird, \quad bird \rightarrow flies$$

is stronger than the theory

$$T_2: \quad bird \vee flies.$$

Here, T_1 derives *bird* and *flies* that are not derived from T_2, so that T_1 is considered more informative than T_2. First-order theories are thus compared and ordered under logical entailment.

Our primary interest in this paper is the corresponding problem in *default logic* [20]. A *default theory* Δ contains *default rules* as well as first-order formulas. Considering comparison of different default theories, the problem is not so simple. For instance, consider the default theory:

$$T_3: \quad bird, \quad \frac{bird : flies}{flies}.$$

Viewing T_1 as a default theory with no default rule, two default theories T_1 and T_3 have the same extension $Th(\{bird, flies\})$. If we compare two theories in terms of formulas derived from each theory, no difference exists between them. Carefully observing each theory, however, the fact *flies* from T_1 is a conclusion derived from first-order formulas, while the same fact from T_3 is a conclusion derived using a default rule. The former is a definite conclusion which is persistent as far as information in the theory is effective, while the latter is a default conclusion possibly withdrawn in face of additional information to the theory. The conclusion *flies* from T_1 is thereby considered stronger than the same conclusion from T_2.

On the other hand, default conclusions are not uniform in default theories. Consider two default theories:

$$T_4: \quad \frac{: innocent}{innocent},$$
$$T_5: \quad \frac{: \neg guilty}{innocent}, \quad \frac{: \neg innocent}{guilty},$$

where T_4 has the single extension $Th(\{innocent\})$, and T_5 has two extensions $Th(\{innocent\})$ and $Th(\{guilty\})$. Then, *innocent* is a *skeptical* default conclusion of T_4, while it is a *credulous* default conclusion of T_5. In this case, T_4 is considered stronger than T_5 in the sense that T_4 has no ambiguity in its conclusion.

From these examples, we can see that a default theory contains different sorts of information in general. Information entailed by the classical portion of a default theory is persistent, while those entailed by the default portion are tentative. On the other hand,

default information that belongs to every extension has strong support to believe, but those belong to some (but not every) extensions are weak. This is a unique feature of default logic which is in contrast to the case of first-order logic where every formula is uniform and has equal position. Those different sorts of information are to be distinguished to compare and order default theories. Such consideration is meaningful and important with the following reasons.

- *Distinguishing different sorts of information*

Studies in nonmonotonic logics have been centered on answering the question: "What information is concluded from a theory (with common-sense)?" On the other hand, few studies answer the question: "What sort of information is concluded from a theory?" Since default theories contain definite and skeptical/credulous default information, distinguishing different sorts of information is meaningful to assess the information content of a theory. Default theories contain *incomplete* information, so that the assessment provides a theoretical ground to measure the degree of "incompleteness" of a theory. Distinction between definite and default consequences derived from a default theory has originally been considered by Ginsberg [13]. He addresses the merit of such distinction as:

> "it should be necessary merely to record the fact that the conclusion never achieved more than default status. ... The default value explicitly admits to the possibility of new information overturning the tentative conclusion it represents".

Such "bookkeeping" mechanism is useful to know "how true or false a given statement is believed to be?" or "how much or little is known about it?" [13]. However, as will be discussed in Section 4.2, Ginsberg does not distinguish skeptical and credulous default consequences derived from a theory.

In a different context, Russell and Norvig [21, p. 360] argue that

> "how can beliefs that have default status be used to make decisions? This is probably the hardest issue for default reasoning. Decisions often involve trade-offs, and one therefore needs to compare the *strength* of belief in the outcomes of different actions".

In this respect, distinguishing different sorts of information would help to compare the strength of belief in the outcomes of different default theories.

- *Comparison of different default theories*

Comparison of theories is intended to know the relative value between them. A theory is considered more valuable than another theory if the former contains more information than the latter. This is because a more informative theory has a greater possibility of solving a problem than a less informative one. Comparison of theories is especially important when there exist multiple sources of information as in *multi-agent systems*. Under the circumstance, an agent having more information could take precedence in problem solving. In first-order logic, a stronger theory is more informative, and theories are ordered by logical entailment. In default logic, however, extensions of theories are not necessarily helpful for judging relative strength between theories as seen in the introductory example. Then, how one can say a theory is stronger than another theory in the context of

default logic? To know the relation, it is necessary to provide a better ability of comparing default theories beyond their extensions. We consider that a default theory is stronger if it brings "more certain" information. In other words, if a default theory Δ_1 is stronger than another theory Δ_2, we have more reason to believe in consequences from Δ_1 than those from Δ_2.

- *Application to logic programming*

Default theories have close connection to *nonmonotonic logic programs*. Nonmonotonic logic programs extend classical Horn logic programs by the introduction of *negation as failure*, and provide a powerful tool for representing and reasoning with incomplete information [3,7]. As nonmonotonic logic programs are considered a subclass of default theories, techniques of ordering default theories are directly applied to the problem of ordering logic programs. Ordering logic programs has an important application. In first-order logic, a theory is called *more general* than another theory if the former is stronger than the latter. Generality relations over first-order clauses have been extensively studied in the fields of *machine learning* and *inductive logic programming* (ILP) [19]. In these fields, generalization is used as a basic operation for inductive learning. Induction problems assume a background theory which is incomplete, otherwise there is no need to learn. However, the present ILP systems mostly handle Horn logic programs as background theories, which are less expressive for representing and reasoning with incomplete knowledge. This leads to the need of constructing a theory of *nonmonotonic inductive logic programming* (NMILP) [22]. Like ILP an induction task in NMILP is to find a new program which generalizes a background theory to account for given examples. In contrast to the classical ILP, a background theory and an induced program are possibly nonmonotonic theories. To construct induction systems that learn nonmonotonic theories, it is necessary to extend the generalization operation and to build a theory for ordering nonmonotonic theories. Ordering default theories and nonmonotonic logic programs thus has potential application to the theory of induction in nonmonotonic logics and nonmonotonic inductive logic programming.

With these background and motivation, this paper studies methods for ordering default theories and nonmonotonic logic programs. To this end, we first provide a multi-valued interpretation for default theories based on a ten-valued *bilattice*. It can distinguish different sorts of information derived from default theories. We then introduce an order relation over default theories, which orders different default theories based on multi-valued interpretations of formulas. Next, we apply the technique to nonmonotonic logic programming, and order extended logic programs under the *answer set semantics*. The *order-equivalence* relation is also introduced as a relation which presents equivalence between theories based on their information contents. Finally, we discuss an application to inductive logic programming.

This is a revised and extended version of the paper [23]. The rest of this paper is organized as follows. Section 2 develops a theory of ordering default theories. Section 3 applies the technique to ordering nonmonotonic logic programs. Section 4 discusses related issues and applications to inductive logic programming. Section 5 summarizes the paper.

2. Ordering default theories

2.1. Default logic

We first review the framework of default logic [20]. A *default theory* is defined as a pair $\Delta = (D, W)$ where D is a set of default rules and W is a set of quantifier-free formulas (called *facts*). A default rule (or simply *default*) is of the form:

$$\frac{\alpha : \beta_1, \ldots, \beta_n}{\gamma},$$

where α, $\beta_1, \ldots, \beta_n$ and γ are quantifier-free formulas and, respectively, called the *prerequisite*, the *justifications* and the *consequent*. In this paper, any default is assumed to have at least one justification ($n \geqslant 1$). A default theory is called *super-normal* if every default is of the form : γ/γ. As defaults and facts are syntactically distinguishable, we often write a default theory Δ as a set $W \cup D$ as far as no confusion arises. Any default/fact with variables represents the set of its ground instances over the Herbrand universe of Δ. Throughout this paper we assume a default theory which is already ground-instantiated, i.e., for any default theory (D, W), D and W contain no variable. Also, a formula means a propositional formula unless stated otherwise. We write $W \models F$ if a formula F is a logical consequence of W. $W_1 \models W_2$ means that every formula which is a logical consequence of W_2 is also a logical consequence of W_1. We write $W_1 \equiv W_2$ iff $W_1 \models W_2$ and $W_2 \models W_1$.

A set S of formulas is *deductively closed* if $S = Th(S)$ where Th is the deductive closure operator as usual. A set E of formulas is an *extension* of (D, W) if it coincides with the smallest deductively closed set E' of formulas satisfying the conditions: (i) $W \subseteq E'$, and (ii) for any ground default $\alpha : \beta_1, \ldots, \beta_n/\gamma$ from D, $\alpha \in E'$ and $\neg\beta_i \notin E$ $(i = 1, \ldots, n)$ imply $\gamma \in E'$. A default theory may have none, one or multiple extensions in general. The set of all extensions of Δ is written as $\mathcal{EXT}(\Delta)$. Given a default theory Δ, a formula is a *credulous* conclusion of Δ if it belongs to some (but not all) extensions. By contrast, a formula is a *skeptical* conclusion of Δ if it belongs to all extensions. [1] An extension E is *inconsistent* if it is the set of all formulas in the language. If a default theory has an inconsistent extension, then this is its only extension.

Proposition 2.1 (*Reiter [20, Corollary 2.2]*). [2] *A default theory $\Delta = (D, W)$ has the inconsistent extension iff W is inconsistent.*

2.2. Multi-valued interpretation of default theories

In classical logic, any formula derived from a theory is a definite consequence of the theory. In default logic, on the other hand, a formula derived from a theory is either a definite consequence by W or a default consequence by D. Moreover, default consequences are brought by two different modes of inferences—skeptical or credulous reasoning. To

[1] In the usual definition, a formula is a credulous conclusion if it belongs to some (possibly every) extension. But in this paper we abuse the term "credulous" to denote *non-skeptical* conclusions.

[2] This property holds for defaults with non-empty justifications.

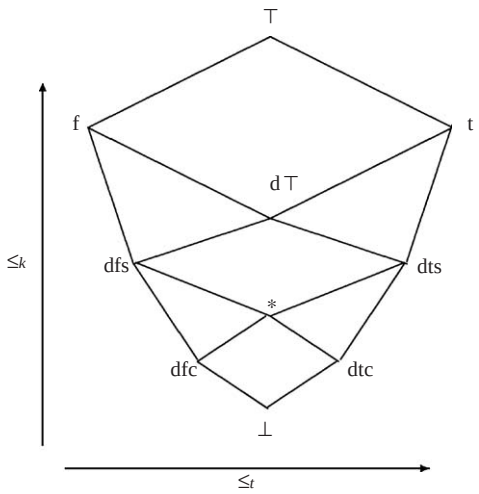

Fig. 1. Bilattice for logic X.

characterize these different types of consequences, we first introduce a multi-valued logic for default reasoning.

Definition 2.1. The logic X has the ten truth values with the meaning as follows—**t**: *true*, **f**: *false*, $\top$: *contradictory*, **d**$\top$: *contradictory by default*, $\bot$: *undefined*, **dts**: *skeptically true by default*, **dfs**: *skeptically false by default*, **dtc**: *credulously true by default*, **dfc**: *credulously false by default*, and $*$: *undetermined by default*.

The truth values of X constitute a *bilattice* under the *knowledge ordering* $\leqslant_k$ and the *truth ordering* $\leqslant_t$ (Fig. 1).[3] Here, it holds that $\bot \leqslant_k \{\mathbf{dtc}, \mathbf{dfc}\} \leqslant_k * \leqslant_k \{\mathbf{dts}, \mathbf{dfs}\} \leqslant_k \mathbf{d}\top \leqslant_k \{\mathbf{t}, \mathbf{f}\} \leqslant_k \top$; $\mathbf{f} \leqslant_t \{\top, \mathbf{d}\top\} \leqslant_t \mathbf{t}$; $\mathbf{dfs} \leqslant_t \mathbf{d}\top \leqslant_t \mathbf{dts}$; and $\mathbf{f} \leqslant_t \mathbf{dfs} \leqslant_t \mathbf{dfc} \leqslant_t \{*, \bot\} \leqslant_t \mathbf{dtc} \leqslant_t \mathbf{dts} \leqslant_t \mathbf{t}$.[4]

The values **t**, **f**, $\top$ are used for the interpretations of definite consequences by W, while the values **dts**, **dfs**, **d**$\top$, **dtc**, **dfc**, $*$ are used for the interpretations of default consequences by D. $\bot$ represents that no information is available. The knowledge ordering reflects the certainty of information content, i.e., first-order logical consequences are more certain than skeptical default consequences, which in turn are more certain than credulous default consequences. By contrast, the truth ordering represents the degree of truth, for instance, "true" has a higher degree of truth than "skeptically true by default", which has a higher degree of truth than "credulously true by default", and so on.

In the logic X, negation $\neg$ is defined as: $\neg\top = \top$, $\neg\mathbf{t} = \mathbf{f}$, $\neg\mathbf{f} = \mathbf{t}$, $\neg\mathbf{dts} = \mathbf{dfs}$, $\neg\mathbf{dfs} = \mathbf{dts}$, $\neg\mathbf{dtc} = \mathbf{dfc}$, $\neg\mathbf{dfc} = \mathbf{dtc}$, $\neg\mathbf{d}\top = \mathbf{d}\top$, $\neg* = *$, $\neg\bot = \bot$, and $\neg\neg x = x$ for

[3] In [23], we used a nine-valued logic and a bilattice which do not have the value **d**$\top$.

[4] $x \leqslant_k \{y, z\}$ means $x \leqslant_k y$ and $x \leqslant_k z$; and $\{x, y\} \leqslant_k z$ means $x \leqslant_k z$ and $y \leqslant_k z$. The same abbreviation is used for $\leqslant_t$.

any $x \in X$. On the other hand, the disjunction $\vee$ and the conjunction $\wedge$ are, respectively, defined by the join operation and the meet operation with respect to the truth ordering in the bilattice. That is, $\mathbf{t} \vee x = \mathbf{t}$ for $x \in X$, $* \vee \bot = \mathbf{dtc}$, $\top \wedge \mathbf{dfc} = \mathbf{f}$, and so on. It is easily verified that $\vee$ and $\wedge$ are associative, commutative, idempotent and absorptive. [5] Note that the truth-functional operations $\vee$ and $\wedge$ have their meaning supplied by the $\leqslant_t$ ordering, while we later order default theories by the $\leqslant_k$ ordering. [6]

Under the logic X the interpretation of a formula in a default theory is defined as follows.

Definition 2.2. Given a default theory $\Delta = (D, W)$, the mapping ϕ_Δ associates a propositional formula F with a truth value of X as follows:

If $\mathcal{EXT}(\Delta) = \emptyset$, $\phi_\Delta(F) = \mathbf{d}\top$ for any formula F; Otherwise,

$$\phi_\Delta(F) = \begin{cases} \top \quad \text{if } W \models F \wedge \neg F; \\ \mathbf{t} \quad \text{if } W \models F \text{ and } W \not\models \neg F; \\ \mathbf{f} \quad \text{if } W \not\models F \text{ and } W \models \neg F; \\ \mathbf{dts} \quad \text{if } W \not\models F,\ W \not\models \neg F, \text{ and } \forall E \in \mathcal{EXT}(\Delta)\ F \in E; \\ \mathbf{dfs} \quad \text{if } W \not\models F,\ W \not\models \neg F, \text{ and } \forall E \in \mathcal{EXT}(\Delta)\ \neg F \in E; \\ \mathbf{dtc} \quad \text{if } \exists E \in \mathcal{EXT}(\Delta) \text{ s.t. } F \in E, \\ \qquad \exists E' \in \mathcal{EXT}(\Delta) \text{ s.t. } F \notin E', \\ \qquad \text{and } \forall E'' \in \mathcal{EXT}(\Delta)\ \neg F \notin E''; \\ \mathbf{dfc} \quad \text{if } \exists E \in \mathcal{EXT}(\Delta) \text{ s.t. } \neg F \in E, \\ \qquad \exists E' \in \mathcal{EXT}(\Delta) \text{ s.t. } \neg F \notin E', \\ \qquad \text{and } \forall E'' \in \mathcal{EXT}(\Delta)\ F \notin E''; \\ * \quad \text{if } \exists E \in \mathcal{EXT}(\Delta) \text{ s.t. } F \in E, \\ \qquad \exists E' \in \mathcal{EXT}(\Delta) \text{ s.t. } \neg F \in E', \\ \qquad \text{and } \forall E'' \in \mathcal{EXT}(\Delta)\ F \wedge \neg F \notin E''; \\ \bot \quad \text{if } \forall E \in \mathcal{EXT}(\Delta)\ F \notin E \text{ and } \neg F \notin E. \end{cases}$$

The mapping ϕ_Δ provides multi-valued interpretations of formulas in a default theory. Intuitively, $\phi_\Delta(F) \in \{\mathbf{t}, \mathbf{f}\}$ means that F or $\neg F$ is a definite conclusion from W. When $\phi_\Delta(F) = \top$ for some formula F, $\phi_\Delta(G) = \top$ for any formula G. This is because in this case W is inconsistent and entails every formula. [7] On the other hand, when $\phi_\Delta(F) \in \{\mathbf{dts}, \mathbf{dfs}\}$ (resp., $\phi_\Delta(F) \in \{\mathbf{dtc}, \mathbf{dfc}\}$), F or $\neg F$ is a default conclusion inferred skeptically (resp., credulously) from Δ. When $\phi_\Delta(F) = *$, a formula F belongs to some extension and its negation $\neg F$ belongs to another extension. In this case, the truth value of F is undetermined by default. Remark that a formula $F \wedge \neg F$ is included in an extension E of Δ iff W is inconsistent (Proposition 2.1). ϕ_Δ maps a formula F into $\bot$ when $\mathcal{EXT}(\Delta) \neq \emptyset$ and F is included in no extension.

---

[5] They are not distributive, e.g., $* \vee (\top \wedge \bot) \neq (* \vee \top) \wedge (* \vee \bot)$.

[6] The join operation and the meet operation with respect to the knowledge ordering are defined as in [11,13]. But those operations are not used for ordering theories, so we do not introduce them in this paper.

[7] In this sense, our logic is not "paraconsistent" (cf. Section 4.2).

With this mapping, the unique truth value from X is assigned to every formula in any default theory. Thus, every default theory obtains the single meaning even when it has no extension or multiple extensions.

Example 2.1. Let Δ be the theory:

$$bird, \quad \frac{bird : flies}{flies},$$

which has the single extension $Th(\{bird, flies\})$. Then $\phi_\Delta(bird) = \mathbf{t}$, $\phi_\Delta(flies) = \mathbf{dts}$, and $\phi_\Delta(bird \to flies) = \mathbf{dts}$, etc.

Example 2.2. Let Δ be the theory:

$$\frac{: \neg rh\text{-}broken \wedge lh\text{-}broken}{lh\text{-}broken}, \quad \frac{: \neg lh\text{-}broken \wedge rh\text{-}broken}{rh\text{-}broken},$$

which has two extensions $Th(\{lh\text{-}broken\})$ and $Th(\{rh\text{-}broken\})$. Then $\phi_\Delta(lh\text{-}broken) = \phi_\Delta(rh\text{-}broken) = \mathbf{dtc}$, $\phi_\Delta(lh\text{-}broken \vee rh\text{-}broken) = \mathbf{dts}$ and $\phi_\Delta(lh\text{-}broken \wedge rh\text{-}broken) = \bot$.

Example 2.3. Let Δ be the theory:

$$quaker \wedge republican, \quad \frac{quaker : pacifist}{pacifist}, \quad \frac{republican : \neg pacifist}{\neg pacifist},$$

which has two extensions $Th(\{quaker \wedge republican, pacifist\})$ and $Th(\{quaker \wedge republican, \neg pacifist\})$. Then, $\phi_\Delta(quaker \wedge republican) = \mathbf{t}$ and $\phi_\Delta(pacifist) = *$.

The followings are some properties of ϕ_Δ.

Proposition 2.2. *For formulas F and G, $F \equiv G$ implies $\phi_\Delta(F) = \phi_\Delta(G)$.*

Proposition 2.3. *For formulas F and G,*

(i) $\phi_\Delta(\neg F) = \neg\phi_\Delta(F)$.
(ii) $\phi_\Delta(F) \geqslant_k \phi_\Delta(G)$ *iff* $\phi_\Delta(\neg F) \geqslant_k \phi_\Delta(\neg G)$.
(iii) $\phi_\Delta(F) \geqslant_t \phi_\Delta(G)$ *iff* $\phi_\Delta(\neg G) \geqslant_t \phi_\Delta(\neg F)$.
(iv) $\neg(\phi_\Delta(F) \vee \phi_\Delta(G)) = \neg\phi_\Delta(F) \wedge \neg\phi_\Delta(G)$.
(v) $\neg(\phi_\Delta(F) \wedge \phi_\Delta(G)) = \neg\phi_\Delta(F) \vee \neg\phi_\Delta(G)$.

Proof. The results of (i)–(iii) immediately hold by the definition of ϕ_Δ and the property of negation in X. De Morgan's laws (iv) and (v) are also verified by the definitions of $\vee$ and $\wedge$. □

Proposition 2.4. *For formulas F and G,*

(i) $\phi_\Delta(F \vee G) \geqslant_t \phi_\Delta(F) \vee \phi_\Delta(G)$.
(ii) $\phi_\Delta(F \wedge G) \leqslant_t \phi_\Delta(F) \wedge \phi_\Delta(G)$.
(iii) $\phi_\Delta(F \to G) \geqslant_t \phi_\Delta(\neg F) \vee \phi_\Delta(G)$.

Proof. (i) By $\phi_\Delta(F \vee G) \geqslant_t \phi_\Delta(F)$ and $\phi_\Delta(F \vee G) \geqslant_t \phi_\Delta(G)$, $\phi_\Delta(F \vee G) \geqslant_t \phi_\Delta(F) \vee \phi_\Delta(G)$ holds. (ii) $\phi_\Delta(F \wedge G) \leqslant_t \phi_\Delta(F)$ and $\phi_\Delta(F \wedge G) \leqslant_t \phi_\Delta(G)$ imply $\phi_\Delta(F \wedge G) \leqslant_t \phi_\Delta(F) \wedge \phi_\Delta(G)$. The result of (iii) follows by the relation $\phi_\Delta(F \to G) = \phi_\Delta(\neg F \vee G)$. □

Example 2.4. In Example 2.2, $\phi_\Delta(lh\text{-}broken \vee rh\text{-}broken) \geqslant_t \phi_\Delta(lh\text{-}broken) \vee \phi_\Delta(rh\text{-}broken)$ and $\phi_\Delta(lh\text{-}broken \wedge rh\text{-}broken) \leqslant_t \phi_\Delta(lh\text{-}broken) \wedge \phi_\Delta(rh\text{-}broken)$.

As shown above, the degree of truth of $\phi_\Delta(F \vee G)$ is generally higher than that of $\phi_\Delta(F) \vee \phi_\Delta(G)$. This reflects the intuition that a disjunction is more likely to hold even when each disjunct is not individually included in an extension. By contrast, the degree of truth of $\phi_\Delta(F \wedge G)$ is generally lower than that of $\phi_\Delta(F) \wedge \phi_\Delta(G)$. This is because a conjunction is less likely to hold even when each conjunct is separately included in an extension. On the other hand, there are no general relations between $\phi_\Delta(F \vee G)$ and $\phi_\Delta(F) \vee \phi_\Delta(G)$; and $\phi_\Delta(F \wedge G)$ and $\phi_\Delta(F) \wedge \phi_\Delta(G)$ under the $\leqslant_k$ ordering.

2.3. Ordering default theories

Based on multi-valued interpretations, we introduce an order relation between default theories.

Definition 2.3. Let Δ_1 and Δ_2 be two default theories which have the same underlying language. Then, Δ_1 is *stronger* than Δ_2 (written as $\Delta_2 \leqslant_{DL} \Delta_1$) if $\phi_{\Delta_2}(F) \leqslant_k \phi_{\Delta_1}(F)$ for any formula F in the language. We write $\Delta_1 \simeq_{DL} \Delta_2$ (called *order-equivalent*) if $\Delta_1 \leqslant_{DL} \Delta_2$ and $\Delta_2 \leqslant_{DL} \Delta_1$.

When $\Delta_2 \leqslant_{DL} \Delta_1$, we also say that Δ_2 is *weaker* than Δ_1. The relation $\leqslant_{DL}$ is a pre-order, i.e., a reflexive and transitive relation on the set of all default theories in the language. Throughout the paper, when we compare different default theories, we assume that they have the same underlying language and the same Herbrand universe.

Intuitively, a default theory Δ_1 is stronger than another default theory Δ_2 if Δ_1 contains at least as much information as Δ_2. In other words, when $\Delta_2 \leqslant_{DL} \Delta_1$, conclusions derived from Δ_1 are relatively more certain and stable than those derived from Δ_2.

Example 2.5. Let Δ_1 and Δ_2 be two default theories:

$$\Delta_1 : bird, \quad penguin, \quad \frac{bird : flies}{flies}, \quad \frac{penguin : \neg flies}{\neg flies},$$

$$\Delta_2 : bird, \quad penguin, \quad \frac{bird : flies \wedge \neg penguin}{flies}, \quad \frac{penguin : \neg flies}{\neg flies},$$

where Δ_1 has two extensions: $Th(\{bird, penguin, flies\})$ and $Th(\{bird, penguin, \neg flies\})$; and Δ_2 has the single extension $Th(\{bird, penguin, \neg flies\})$. Then, $\phi_{\Delta_1}(bird) = \phi_{\Delta_2}(bird) = \mathbf{t}$ and $\phi_{\Delta_1}(penguin) = \phi_{\Delta_2}(penguin) = \mathbf{t}$, while $\phi_{\Delta_1}(flies) = *$ and $\phi_{\Delta_2}(flies) = \mathbf{dfs}$. So, $\Delta_1 \leqslant_{DL} \Delta_2$.

The "stronger" relation reduces to the corresponding relation between (propositional) first-order theories when default theories have no defaults.

Proposition 2.5. *Let* $\Delta_1 = (\emptyset, W_1)$ *and* $\Delta_2 = (\emptyset, W_2)$ *be two default theories. Then,* $\Delta_2 \leqslant_{DL} \Delta_1$ *iff* $W_1 \models W_2$.

Proof. When $\Delta = (\emptyset, W)$, $\phi_\Delta(F)$ takes one of the values $\top$, $\mathbf{t}$, $\mathbf{f}$, $\bot$ for any formula F. In case of $\phi_{\Delta_2}(F) = \top$, $\Delta_2 \leqslant_{DL} \Delta_1$ implies $\phi_{\Delta_1}(F) = \top$. In case of $\phi_{\Delta_2}(F) = \mathbf{t}$ (resp., $\phi_{\Delta_2}(F) = \mathbf{f}$) $\Delta_2 \leqslant_{DL} \Delta_1$ implies $\mathbf{t} \leqslant_k \phi_{\Delta_1}(F)$ (resp., $\mathbf{f} \leqslant_k \phi_{\Delta_1}(F)$). In case of $\phi_{\Delta_2}(F) = \bot$, $\bot \leqslant_k \phi_{\Delta_1}(F)$ holds. In each case $\Delta_2 \leqslant_{DL} \Delta_1$ implies $W_1 \models W_2$. The converse is straightforward. □

Thus, the relation $\leqslant_{DL}$ is a natural extension of the "stronger" relation in (propositional) first-order theories. The above proposition also implies that in first-order theories the order-equivalence relation reduces to the logical equivalence.

Corollary 2.6. *Let* $\Delta_1 = (\emptyset, W_1)$ *and* $\Delta_2 = (\emptyset, W_2)$ *be two default theories. Then,* $\Delta_1 \simeq_{DL} \Delta_2$ *iff* $W_1 \equiv W_2$.

On the other hand, when a default theory contains default rules, the order-equivalence relation is generally stronger than the equivalence based on extensions.

Proposition 2.7. *For two default theories* $\Delta_1 = (D_1, W_1)$ *and* $\Delta_2 = (D_2, W_2)$, $\Delta_1 \simeq_{DL} \Delta_2$ *implies* $\mathcal{EXT}(\Delta_1) = \mathcal{EXT}(\Delta_2)$. *The converse also holds if* $W_1 \equiv W_2$.

Proof. When $\Delta_1 \simeq_{DL} \Delta_2$, $\phi_{\Delta_1}(F) = \phi_{\Delta_2}(F)$ for any formula F. Suppose that Δ_1 has an extension E which is not an extension of Δ_2. If $E = \emptyset$, Δ_1 has the unique extension $\emptyset$ and $\phi_{\Delta_1}(F) = \bot$ for every formula F. By $\Delta_1 \simeq_{DL} \Delta_2$, Δ_2 also has the unique extension $E = \emptyset$. This contradicts the assumption. Else if $E \neq \emptyset$, suppose a formula $G = \bigwedge_{F \in E} F$. When Δ_2 has no extension that includes G, $\phi_{\Delta_1}(G) \neq \phi_{\Delta_2}(G)$. Contradiction. When Δ_2 has an extension E' that includes G, $E \subseteq E'$ holds. Then, for any formula $F' \in E' \setminus E$, put $G' = G \wedge F'$. Since Δ_1 has no extension which includes G', $\phi_{\Delta_1}(G') \neq \phi_{\Delta_2}(G')$. Again, contradiction. Hence, $\mathcal{EXT}(\Delta_1) = \mathcal{EXT}(\Delta_2)$.

To see the converse, let $W_1 \equiv W_2$. If $\mathcal{EXT}(\Delta_1) = \mathcal{EXT}(\Delta_2) = \emptyset$, $\phi_{\Delta_1}(F) = \phi_{\Delta_2}(F) = \mathbf{d}\top$ for any formula F. Else if $\mathcal{EXT}(\Delta_1) = \mathcal{EXT}(\Delta_2) \neq \emptyset$, $\phi_{\Delta_1}(F) = \mathbf{t}$ (resp., $\mathbf{f}$, $\top$) iff $\phi_{\Delta_2}(F) = \mathbf{t}$ (resp., $\mathbf{f}$, $\top$) for any formula F. By $\mathcal{EXT}(\Delta_1) = \mathcal{EXT}(\Delta_2)$ and the definition of the mapping ϕ_Δ, it is easy to see that $\phi_{\Delta_1}(F) = \phi_{\Delta_2}(F) = x$ for $x \in \{\mathbf{dts}, \mathbf{dfs}, \mathbf{dtc}, \mathbf{dfc}, *, \bot\}$. Hence, $\Delta_1 \simeq_{DL} \Delta_2$. □

Example 2.6. Consider the following three default theories:

$$\Delta_1 : \mathit{bird}, \quad \mathit{bird} \to \mathit{flies},$$
$$\Delta_2 : \mathit{bird}, \quad \frac{\mathit{bird} : \mathit{flies}}{\mathit{flies}},$$
$$\Delta_3 : \mathit{bird}, \quad \frac{: \mathit{bird} \to \mathit{flies}}{\mathit{bird} \to \mathit{flies}}.$$

Then, $\Delta_2 \leqslant_{DL} \Delta_1$ and $\Delta_3 \leqslant_{DL} \Delta_1$, while $\Delta_2 \simeq_{DL} \Delta_3$. Note that all Δ_1, Δ_2 and Δ_3 have the same extension $Th(\{bird, flies\})$.

When two default theories are order-equivalent, it means that they not only have the same extensions but also share the same sort of information. Thus, the order-equivalence relation provides an equivalence with finer granularity with regard to the information contents of default theories.

The order $\leqslant_{DL}$ is nonmonotonic with respect to the increase of information.

Proposition 2.8. *Let Δ_1 and Δ_2 be two default theories and F a formula. Then, $\Delta_1 \leqslant_{DL} \Delta_2$ implies neither $\Delta_1 \leqslant_{DL} \Delta_2 \cup \{F\}$ nor $\Delta_1 \cup \{F\} \leqslant_{DL} \Delta_2 \cup \{F\}$. In particular, $\Delta_1 \not\leqslant_{DL} \Delta_1 \cup \{F\}$ in general.*

Example 2.7. Let Δ_1 and Δ_2 be two default theories:

$$\Delta_1 : \frac{: p \wedge \neg q}{p},$$
$$\Delta_2 : \frac{: p \wedge \neg q}{p}, \quad r,$$

where $\phi_{\Delta_1}(p) = \phi_{\Delta_2}(p) = \mathbf{dts}$, $\phi_{\Delta_1}(q) = \phi_{\Delta_2}(q) = \bot$, $\phi_{\Delta_1}(r) = \bot$ and $\phi_{\Delta_2}(r) = \mathbf{t}$. Then, $\Delta_1 \leqslant_{DL} \Delta_2$ holds. Let $F = (r \to q)$. Then, $\Delta_1 \not\leqslant_{DL} \Delta_2 \cup \{F\}$ and $\Delta_1 \cup \{F\} \not\leqslant_{DL} \Delta_2 \cup \{F\}$ by $\phi_{\Delta_1 \cup \{F\}}(p) = \mathbf{dts}$ and $\phi_{\Delta_2 \cup \{F\}}(p) = \bot$.

The introduction of new information may block the application of some default rules, which would cause the withdrawal of some default conclusions in a theory. This is a typical feature of default reasoning.

Example 2.8. Let Δ_1, Δ_2 and Δ_3 be three default theories:

$$\Delta_1 : \frac{: \neg p}{q}, \quad \frac{: \neg q}{p},$$
$$\Delta_2 : \frac{p : \neg q}{\neg q}, \quad \frac{q : \neg p}{\neg p},$$
$$\Delta_3 : \frac{: p \wedge q}{p \wedge q}, \quad \frac{: \neg p}{\neg p},$$

where $\phi_{\Delta_1}(p) = \phi_{\Delta_1}(q) = \mathbf{dtc}$, $\phi_{\Delta_2}(p) = \phi_{\Delta_2}(q) = \bot$, $\phi_{\Delta_3}(p) = *$ and $\phi_{\Delta_3}(q) = \mathbf{dtc}$. Then, $\Delta_1 \leqslant_{DL} \Delta_3$ and $\Delta_2 \leqslant_{DL} \Delta_3$, while $\Delta_3 \leqslant_{DL} \Delta_1 \cup \Delta_2$.

The above example shows that $\Delta_1 \leqslant_{DL} \Delta_3$ and $\Delta_2 \leqslant_{DL} \Delta_3$ do not generally imply $\Delta_1 \cup \Delta_2 \leqslant_{DL} \Delta_3$. Likewise, $\Delta_1 \leqslant_{DL} \Delta_2$ and $\Delta_1 \leqslant_{DL} \Delta_3$ do not generally imply $\Delta_1 \leqslant_{DL} \Delta_2 \cup \Delta_3$. This means that collaborating weaker theories often produce a much stronger theory, and combining stronger theories does not always produce a much stronger theory. In particular, $\Delta_1 \leqslant_{DL} \Delta_2$ implies neither $\Delta_1 \leqslant_{DL} \Delta_1 \cup \Delta_2$ nor $\Delta_1 \cup \Delta_2 \leqslant_{DL} \Delta_2$.

We finally provide a connection between the order relation $\leqslant_{DL}$ and default extensions.

Theorem 2.9. *Let $\Delta_1 = (D_1, W_1)$ and $\Delta_2 = (D_2, W_2)$ be two default theories. Then, $\Delta_1 \leqslant_{DL} \Delta_2$ if the following conditions are satisfied*:

1. $W_2 \models W_1$,
2. $\forall E_2 \in \mathcal{EXT}(\Delta_2)\ \exists E_1 \in \mathcal{EXT}(\Delta_1)$ s.t. $E_1 \subseteq E_2$,
3. $\forall E_1 \in \mathcal{EXT}(\Delta_1)\ \exists E_2 \in \mathcal{EXT}(\Delta_2)$ s.t. $E_1 \subseteq E_2$.

Proof. Let F be any formula. First, suppose $\mathcal{EXT}(\Delta_1) = \emptyset$. By the second and third conditions, $\mathcal{EXT}(\Delta_1) = \emptyset$ iff $\mathcal{EXT}(\Delta_2) = \emptyset$. In this case, $\phi_{\Delta_1}(F) = \phi_{\Delta_2}(F) = \mathbf{d}\top$, thereby $\Delta_1 \leqslant_{DL} \Delta_2$. Next, suppose $\mathcal{EXT}(\Delta_1) \neq \emptyset$. In case of $\phi_{\Delta_1}(F) = \top$, the first condition implies $\phi_{\Delta_2}(F) = \top$. In case of $\phi_{\Delta_1}(F) = \mathbf{t}$ (resp., $\mathbf{f}$), the first condition implies $\phi_{\Delta_1}(F) = \mathbf{t}$ (resp., $\mathbf{f}$). In case of $\phi_{\Delta_1}(F) = \mathbf{dts}$, $\phi_{\Delta_1}(F) \notin \{\top, \mathbf{t}, \mathbf{f}\}$ and F is included in every extension of Δ_1. By the second condition, F is included in every extension of Δ_2. Then, $\mathbf{dts} \leqslant_k \phi_{\Delta_2}(F)$. Similarly, it is shown that $\phi_{\Delta_1}(F) = \mathbf{dfs}$ implies $\mathbf{dfs} \leqslant_k \phi_{\Delta_2}(F)$. In case of $\phi_{\Delta_1}(F) = \mathbf{dtc}$, F is included in some (but not every) extension of Δ_1. By the third condition, F is included in some extension of Δ_2. Then, $\mathbf{dtc} \leqslant_k \phi_{\Delta_2}(F)$. Similarly, it is shown that $\phi_{\Delta_1}(F) = \mathbf{dfc}$ implies $\mathbf{dfc} \leqslant_k \phi_{\Delta_2}(F)$. In case of $\phi_{\Delta_1}(F) = *$, the third condition implies either $\phi_{\Delta_2}(F) = *$ or $\phi_{\Delta_2}(F) = \top$. Thus, $* \leqslant_k \phi_{\Delta_2}(F)$. In case of $\phi_{\Delta_1}(F) = \bot$, it holds that $\bot \leqslant_k \phi_{\Delta_2}(F)$. Hence, in every case $\Delta_1 \leqslant_{DL} \Delta_2$ holds. □

Theorem 2.9 provides a sufficient condition to see $\Delta_1 \leqslant_{DL} \Delta_2$ using extensions of default theories. For a necessary condition, we have the following result for a restricted case.

Theorem 2.10. *Let $\Delta_1 = (D_1, W_1)$ and $\Delta_2 = (D_2, W_2)$ be two default theories. When $\mathcal{EXT}(\Delta_1) \neq \emptyset$, $\Delta_1 \leqslant_{DL} \Delta_2$ implies $W_2 \models W_1$.*

Proof. For any formula F, when $\phi_{\Delta_1}(F) = \top$, $\Delta_1 \leqslant_{DL} \Delta_2$ implies $\phi_{\Delta_2}(F) = \top$. Else when $\phi_{\Delta_1}(F) = \mathbf{t}$ (resp., $\mathbf{f}$), $\Delta_1 \leqslant_{DL} \Delta_2$ implies $\phi_{\Delta_2}(F) \in \{\top, \mathbf{t}\}$ (resp., $\phi_{\Delta_2}(F) \in \{\top, \mathbf{f}\}$). Thus, in each case $W_2 \models W_1$. On the other hand, when $\phi_{\Delta_1}(F) \in \{\mathbf{dts}, \mathbf{dfs}, \mathbf{dtc}, \mathbf{dfc}, *, \bot\}$, $W_1 \not\models F$. Hence, $W_2 \models W_1$. □

When $\mathcal{EXT}(\Delta_1) = \emptyset$, the above implication does not hold in general.

Example 2.9. Let Δ_1 and Δ_2 be two default theories:

$$\Delta_1 : p, \quad \frac{: \neg q}{q},$$
$$\Delta_2 : \neg p \wedge q.$$

Then, $\Delta_1 \leqslant_{DL} \Delta_2$ but $W_2 \not\models W_1$.

3. Ordering nonmonotonic logic programs

3.1. Extended logic programs

In logic programming, default reasoning is realized by *negation as failure* (NAF). Logic programs containing NAF are called *nonmonotonic logic programs*.

Nonmonotonic logic programs considered in this paper are the class of *extended logic programs* (ELPs) [12], which contain two kinds of negation; explicit (or classical) negation $\neg$ and NAF (or default negation) *not*. An extended logic program (or simply a program) is a set of *rules* of the form:

$$L_0 \leftarrow L_1, \ldots, L_m, \mathit{not}\ L_{m+1}, \ldots, \mathit{not}\ L_n \quad (n \geqslant m),$$

where each L_i $(0 \leqslant i \leqslant n)$ is a positive/negative literal, i.e., A or $\neg A$ with an atom A, and *not* represents NAF. The literal L_0 is the *head* and the conjunction $L_1, \ldots, L_m, \mathit{not}\ L_{m+1}, \ldots, \mathit{not}\ L_n$ is the *body* of the rule. A rule or a program is called *not-free* if it contains no NAF (i.e., $m = n$). Given an ELP Π, the set of *not*-free rules from Π is denoted by Π^+. A rule with the empty body $L \leftarrow$ is identified with the literal L and called a *fact*. The head of any rule is non-empty. [8] A program Π containing variables is semantically identified with its ground instantiation, i.e., the set of ground rules obtained from Π by substituting variables with elements of the Herbrand universe of Π in every possible way. We handle ground programs throughout the paper.

The semantics of ELPs is given by the *answer set semantics* [12]. Let *Lit* be the set of all ground literals in the language of a program (called the *literal base*). Suppose an ELP Π and a set of literals $S(\subseteq Lit)$. Then, the *reduct* Π^S is the program which contains the ground rule $L_0 \leftarrow L_1, \ldots, L_m$ iff there is a rule $L_0 \leftarrow L_1, \ldots, L_m, \mathit{not}\ L_{m+1}, \ldots, \mathit{not}\ L_n$ in the ground instantiation of Π such that $\{L_{m+1}, \ldots, L_n\} \cap S = \emptyset$. Given a *not*-free ELP Π, $Cn(\Pi)$ denotes the smallest set of ground literals which is (i) *closed* under Π, i.e., for every ground rule $L_0 \leftarrow L_1, \ldots, L_m$ from the ground instantiation of Π, $\{L_1, \ldots, L_m\} \subseteq Cn(\Pi)$ implies $L_0 \in Cn(\Pi)$; and (ii) *logically closed*, i.e., it is either consistent or equal to *Lit*. Given an ELP Π and a set S of literals, S is an *answer set* of Π if $S = Cn(\Pi^S)$.

Answer sets represent possible beliefs of a program, and an ELP may have none, one, or multiple answer sets. In particular, every *not*-free ELP Π has the unique answer set $Cn(\Pi)$. An answer set is *consistent* if it is not *Lit*. If a program has the contradictory answer set *Lit*, then this is its only answer set. A program is *consistent* if it has a consistent answer set. The set of all answer sets of an ELP Π is written as $\mathcal{AS}(\Pi)$. A literal is a *credulous* conclusion of a program if it belongs to some (but not all) answer sets of the program; a literal is a *skeptical* conclusion of a program if it belongs to all answer sets of the program. [9]

Proposition 3.1. *An ELP Π has the unique answer set Lit iff $Cn(\Pi^+) = Lit$.*

[8] Under the answer set semantics which we consider in this paper, a rule with the empty head $\leftarrow F$ is expressed by the semantically equivalent rule $L \leftarrow F, \mathit{not}\ L$ with a literal L.

[9] Again we use the term "credulous" conclusions to denote non-skeptical conclusions.

Proof. Π has the answer set Lit iff Π^{Lit} has the answer set Lit. As $\Pi^{Lit} = \Pi^+$, the result follows. □

According to Gelfond and Lifschitz [12], the rule $L_0 \leftarrow L_1, \ldots, L_m, not\ L_{m+1}, \ldots, not\ L_n$ is interpreted as the default rule:

$$\frac{L_1 \wedge \cdots \wedge L_m : \neg L_{m+1}, \ldots, \neg L_n}{L_0},$$

where $\neg\neg L = L$ for a positive literal L. In this case, there is a 1-1 correspondence between the answer sets of a program and the extensions of the corresponding default theory. [10]

Proposition 3.2 (*Gelfond and Lifschitz [12, Proposition 3]*). *Let Π be an ELP and Δ_Π its corresponding default theory. If S is an answer set of Π, then the deductive closure of S is an extension of Δ_Π. Conversely, every extension of Δ_Π is the deductive closure of exactly one answer set of Π.*

3.2. Ordering ELPs

Using the correspondence between an ELP and a default theory, a multi-valued interpretation for ELPs is defined under the logic X.

Definition 3.1. Given an ELP Π, the mapping ϕ_Π associates a literal $L \in Lit$ with a truth value of X as follows:

If $\mathcal{AS}(\Pi) = \emptyset$, $\phi_\Pi(L) = \mathbf{d}\top$ for any literal L; Otherwise,

$$\phi_\Pi(L) = \begin{cases} \top & \text{if } L \in Cn(\Pi^+) \text{ and } \neg L \in Cn(\Pi^+); \\ \mathbf{t} & \text{if } L \in Cn(\Pi^+) \text{ and } \neg L \notin Cn(\Pi^+); \\ \mathbf{f} & \text{if } L \notin Cn(\Pi^+) \text{ and } \neg L \in Cn(\Pi^+); \\ \mathbf{dts} & \text{if } L \notin Cn(\Pi^+),\ \neg L \notin Cn(\Pi^+), \\ & \quad \text{and } \forall S \in \mathcal{AS}(\Pi)\ L \in S; \\ \mathbf{dfs} & \text{if } L \notin Cn(\Pi^+),\ \neg L \notin Cn(\Pi^+), \\ & \quad \text{and } \forall S \in \mathcal{AS}(\Pi)\ \neg L \in S; \\ \mathbf{dtc} & \text{if } \exists S \in \mathcal{AS}(\Pi) \text{ s.t. } L \in S,\ \exists T \in \mathcal{AS}(\Pi) \text{ s.t. } L \notin T, \\ & \quad \text{and } \forall U \in \mathcal{AS}(\Pi)\ \neg L \notin U; \\ \mathbf{dfc} & \text{if } \exists S \in \mathcal{AS}(\Pi) \text{ s.t. } \neg L \in S,\ \exists T \in \mathcal{AS}(\Pi) \text{ s.t. } \neg L \notin T, \\ & \quad \text{and } \forall U \in \mathcal{AS}(\Pi)\ L \notin U; \\ * & \text{if } \exists S \in \mathcal{AS}(\Pi) \text{ s.t. } L \in S,\ \exists T \in \mathcal{AS}(\Pi) \text{ s.t. } \neg L \in T, \\ & \quad \text{and } \forall U \in \mathcal{AS}(\Pi) \text{ either } L \notin U \text{ or } \neg L \notin U; \\ \bot & \text{if } \forall S \in \mathcal{AS}(\Pi)\ L \notin S \text{ and } \neg L \notin S. \end{cases}$$

[10] Precisely speaking, *not*-free rules in an ELP correspond to justification-free defaults. Although we supposed defaults with nonempty justifications in Section 2, the following discussion is valid apart from the results of the previous section.

Note that literals L and $\neg L$ are included in every answer set of Π iff they are in $Cn(\Pi^+)$ (Proposition 3.1).

The intuitive meaning of ϕ_Π is analogous to that of ϕ_Δ. Remark that there is a difference between ϕ_Π and ϕ_Δ on the definition of the values **t**, **f** and $\top$. This is due to the fact that *not*-free rules in $Cn(\Pi^+)$ are interpreted as justification-free defaults in default logic (Proposition 3.2). Thus, the rule $p \leftarrow q$ in Π^+ has a meaning different from the first-order formula $p \vee \neg q$ in W. Accordingly, there is no correspondence between consequences from $Cn(\Pi^+)$ and consequences from W in a default theory.

With the mapping ϕ_Π, every program obtains the unique meaning even when the program has no/multiple answer sets.

Example 3.1. Let Π be the program:

$$p \leftarrow not\ q,$$
$$q \leftarrow not\ p,$$
$$r \leftarrow not\ \neg s,$$

which has two answer sets $\{p, r\}$ and $\{q, r\}$. Then $\phi_\Pi(p) = \phi_\Pi(q) = \mathbf{dtc}$, $\phi_\Pi(r) = \mathbf{dts}$, and $\phi_\Pi(s) = \bot$.

Example 3.2. Let Π be the program:

$$p \leftarrow not\ \neg p,$$
$$\neg p \leftarrow not\ p,$$

which has two answer sets $\{p\}$ and $\{\neg p\}$. Then, $\phi_\Pi(p) = *$.

Example 3.3. Let Π be the program:

$$p \leftarrow not\ q,$$
$$q \leftarrow not\ r,$$
$$r \leftarrow not\ p,$$

which has no answer set. Then, $\phi_\Pi(p) = \phi_\Pi(q) = \phi_\Pi(r) = \mathbf{d}\top$.

ϕ_Π has the properties obtained from Proposition 2.3 by replacing ϕ_Δ with ϕ_Π and formulas with literals.

An order relation between ELPs is defined as follows.

Definition 3.2. Let Π_1 and Π_2 be two ELPs which have the same literal base Lit. Then, Π_1 is *stronger* than Π_2 under the answer set semantics (written as $\Pi_2 \leqslant_{AS} \Pi_1$) if $\phi_{\Pi_2}(L) \leqslant_k \phi_{\Pi_1}(L)$ for any literal $L \in Lit$. We write $\Pi_1 \simeq_{AS} \Pi_2$ (called *order-equivalent*) if $\Pi_1 \leqslant_{AS} \Pi_2$ and $\Pi_2 \leqslant_{AS} \Pi_1$.

The relation $\leqslant_{AS}$ is a pre-order on the set of all ELPs in the language. A program Π_1 is stronger than Π_2 if Π_1 has at least as much information as Π_2. In contrast to default logic, we compare programs in terms of literals included in answer sets. This is because

in nonmonotonic logic programs the meaning of a program is determined by consequent literals included in selected models of a program.

In what follows, when we compare different programs, we assume that they have the same literal base. The relation $\leqslant_{AS}$ has the following properties.

Proposition 3.3. *For two not-free ELPs Π_1 and Π_2, $\Pi_1 \leqslant_{AS} \Pi_2$ iff $Cn(\Pi_1) \subseteq Cn(\Pi_2)$.*

Proof. Not-free ELPs Π_1 and Π_2 have the single answer set $Cn(\Pi_1)$ and $Cn(\Pi_2)$, respectively. Then, $\phi_{\Pi_1}(L)$ and $\phi_{\Pi_2}(L)$ take one of the values $\top$, $\mathbf{t}$, $\mathbf{f}$, $\bot$ for any literal L. In case of $\phi_{\Pi_1}(L) = \top$, $\Pi_1 \leqslant_{AS} \Pi_2$ iff $Cn(\Pi_1) = Cn(\Pi_2) = Lit$. In case of $\phi_{\Pi_1}(L) \in \{\mathbf{t}, \mathbf{f}, \bot\}$, $\Pi_1 \leqslant_{AS} \Pi_2$ iff $Cn(\Pi_1) \subseteq Cn(\Pi_2)$. Hence, the result holds. □

Corollary 3.4. *For two not-free ELPs Π_1 and Π_2, $\Pi_1 \simeq_{AS} \Pi_2$ iff $Cn(\Pi_1) = Cn(\Pi_2)$ iff $\mathcal{AS}(\Pi_1) = \mathcal{AS}(\Pi_2)$.*

Proof. By Proposition 3.3, $\Pi_1 \simeq_{AS} \Pi_2$ iff $Cn(\Pi_1) = Cn(\Pi_2)$. Since $Cn(\Pi_1)$ (resp., $Cn(\Pi_2)$) is the answer set of Π_1 (resp., Π_2), the result holds. □

The order-equivalence relation $\simeq_{AS}$ sometimes provides a stronger relation than the normal equivalence relation based on answer sets.

Proposition 3.5. *Let Π_1 and Π_2 be two ELPs such that each program has at most one answer set. Then, $\Pi_1 \simeq_{AS} \Pi_2$* implies *$\mathcal{AS}(\Pi_1) = \mathcal{AS}(\Pi_2)$. The converse also holds if $Cn(\Pi_1^+) = Cn(\Pi_2^+)$.*

Proof. Let $\Pi_1 \simeq_{AS} \Pi_2$. Then, $\phi_{\Pi_1}(L) = \phi_{\Pi_2}(L)$ for any literal $L \in Lit$. Suppose that Π_1 has no answer set. Then, $\Pi_1 \simeq_{AS} \Pi_2$ implies $\mathcal{AS}(\Pi_1) = \mathcal{AS}(\Pi_2) = \emptyset$. Next, suppose that Π_1 has the single answer set S and Π_2 has the single answer set T. Assume that there is a literal L such that $L \in S \setminus T$ or $L \in T \setminus S$. When $L \in S \setminus T$, $\phi_{\Pi_1}(L) \geqslant_k \mathbf{dts}$ but $\phi_{\Pi_2}(L) = \bot$. This contradicts the assumption $\phi_{\Pi_1}(L) = \phi_{\Pi_2}(L)$. Similarly, contradiction arises when $L \in T \setminus S$. Thus, there is no such literal, thereby $S = T$. Hence, $\mathcal{AS}(\Pi_1) = \mathcal{AS}(\Pi_2)$.

To see the converse, let $Cn(\Pi_1^+) = Cn(\Pi_2^+)$. If $\mathcal{AS}(\Pi_1) = \mathcal{AS}(\Pi_2) = \emptyset$. $\phi_{\Pi_1}(L) = \phi_{\Pi_2}(L) = \mathbf{d}\top$ for any literal $L \in Lit$. Else if $\mathcal{AS}(\Pi_1) = \mathcal{AS}(\Pi_2) = \{S\}$, $\phi_{\Pi_1}(L)$ or $\phi_{\Pi_2}(L)$ takes one of the values $\top$, $\mathbf{t}$, $\mathbf{f}$, $\mathbf{dts}$, $\mathbf{dfs}$, $\bot$ for any literal $L \in Lit$. By $Cn(\Pi_1^+) = Cn(\Pi_2^+)$, $\phi_{\Pi_1}(L) = \mathbf{t}$ (resp., $\mathbf{f}$, $\top$) iff $\phi_{\Pi_2}(L) = \mathbf{t}$ (resp., $\mathbf{f}$, $\top$) for any $L \in Lit$. Then, $\phi_{\Pi_1}(L) = \phi_{\Pi_2}(L) = x$ for $x \in \{\mathbf{dts}, \mathbf{dfs}, \bot\}$. Hence, $\Pi_1 \simeq_{AS} \Pi_2$ holds. □

$\Pi_1 \simeq_{AS} \Pi_2$ does not imply $\mathcal{AS}(\Pi_1) = \mathcal{AS}(\Pi_2)$ in general.

Example 3.4. Let Π_1 and Π_2 be two programs:

$$\begin{aligned} \Pi_1 :\ & p \leftarrow not\ r, \\ & q \leftarrow not\ r, \\ & r \leftarrow not\ q, \end{aligned}$$

$$\begin{aligned}\Pi_2 :\ & p \leftarrow not\ q,\\ & q \leftarrow not\ r,\\ & r \leftarrow not\ q,\end{aligned}$$

where $\mathcal{AS}(\Pi_1) = \{\{p, q\}, \{r\}\}$ and $\mathcal{AS}(\Pi_2) = \{\{p, r\}, \{q\}\}$. On the other hand, all p, q, and r have the value **dtc** in both Π_1 and Π_2, thereby $\Pi_1 \simeq_{AS} \Pi_2$.

Thus, the order-equivalence relation between logic programs provides an equivalence relation which is independent of the normal equivalence relation based on answer sets. Such incompatibility with the case of default theories will be argued in Section 4.3 in depth.

The order $\leqslant_{AS}$ has nonmonotonic properties corresponding to Proposition 2.8 with respect to the introduction of new rules/literals to a program.

A connection between the order relation $\leqslant_{AS}$ and answer sets is given as follows.

Theorem 3.6. *Let Π_1 and Π_2 be two ELPs. Then, $\Pi_1 \leqslant_{AS} \Pi_2$ if the following conditions are satisfied*:

1. $Cn(\Pi_1^+) \subseteq Cn(\Pi_2^+)$,
2. $\forall S \in \mathcal{AS}(\Pi_2)\ \exists T \in \mathcal{AS}(\Pi_1)$ s.t. $T \subseteq S$,
3. $\forall T \in \mathcal{AS}(\Pi_1)\ \exists S \in \mathcal{AS}(\Pi_2)$ s.t. $T \subseteq S$.

In particular, if $\mathcal{AS}(\Pi_1) \neq \emptyset$, $\Pi_1 \leqslant_{AS} \Pi_2$ implies $Cn(\Pi_1^+) \subseteq Cn(\Pi_2^+)$.

Proof. Similar to the proofs of Theorems 2.9 and 2.10. □

3.3. Transformational properties

In logic programming, program transformations are used for optimizing a program while preserving the original meaning of the program. Moreover, any reasonable semantics is required to be robust under elementary program transformations. In this section we present the effect of program transformations on the multi-valued semantics of logic programs. Program transformations considered here are: *unfold/fold* (UNFOLD/FOLD), *elimination of tautologies* (TAUT), *positive/negative reduction* (RED$^+$/RED$^-$), *elimination of nonminimal rules* (NONMIN), and *elimination of contradictions* (CONTRA). These are basic and representative transformations appearing in the literature [7,26].

For a ground rule r of the form: $L_0 \leftarrow L_1, \ldots, L_m, not\ L_{m+1}, \ldots, not\ L_n$, define $H(r) = \{L_0\}$, $B^+(r) = \{L_1, \ldots, L_m\}$ and $B^-(r) = \{L_{m+1}, \ldots, L_n\}$. The rule r is then written as $H(r) \leftarrow B^+(r), not\ B^-(r)$. Using these notations, the above transformations are described as follows: Let Π be a (ground) ELP.

UNFOLD: Replace a rule $H(r) \leftarrow B^+(r), not\ B^-(r)$ in Π with rules $H(r) \leftarrow (B^+(r) \setminus \{L\}) \cup B^+(r_i), not\ (B^-(r) \cup B^-(r_i))$, where $L \in B^+(r)$ and $L \leftarrow B^+(r_i), not\ B^-(r_i)$ are all rules in Π with the head L.

FOLD: Folding is the reverse transformation of unfolding. Replace a rule $H(r) \leftarrow B^+(r), not\ B^-(r)$ in Π with $H(r) \leftarrow L, (B^+(r) \setminus B^+(r')), not\ (B^-(r) \setminus B^-(r'))$ if there is another rule $L \leftarrow B^+(r'), not\ B^-(r')$ in Π such that $B^+(r') \subseteq B^+(r)$ and $B^-(r') \subseteq B^-(r)$, and L appears in the heads of no other rules in Π.

TAUT: Delete a rule r from Π if $H(r) \cap B^+(r) \neq \emptyset$.
RED$^+$: Replace a rule $H(r) \leftarrow B^+(r), not\ B^-(r)$ in Π with $H(r) \leftarrow B^+(r), not\ (B^-(r) \setminus \{L\})$ if L appears in the head of no rule in Π.
RED$^-$: Delete a rule r from Π if there is a fact $L \leftarrow$ in Π such that $L \in B^-(r)$.
NONMIN: Delete a rule r from Π if there is another rule r' in Π such that $H(r) = H(r')$, $B^+(r') \subseteq B^+(r)$ and $B^-(r') \subseteq B^-(r)$.
CONTRA: Delete a rule r from Π if $B^+(r) \cap B^-(r) \neq \emptyset$.

It is known that the above program transformations all preserve the answer sets of an ELP Π [1,7].

Example 3.5. Let Π be the program:

$$p \leftarrow q,\ not\ r,$$
$$q \leftarrow not\ s,$$
$$q \leftarrow p,$$
$$s \leftarrow .$$

Applying **UNFOLD** to the first rule, it becomes

$$p \leftarrow not\ s,\ not\ r,$$
$$p \leftarrow p,\ not\ r,$$
$$q \leftarrow not\ s,$$
$$q \leftarrow p,$$
$$s \leftarrow .$$

The first rule and the third rule are deleted by **RED**$^-$, and the second rule is eliminated by **TAUT**. As a result, the program becomes

$$q \leftarrow p,$$
$$s \leftarrow .$$

Given an ELP Π, let $tr(\Pi)$ be a program which is obtained from Π by applying transformations other than **RED**$^+$. Then, we have the following results.

Theorem 3.7. *Let Π be an ELP. Then, $\phi_\Pi(L) = \phi_{tr(\Pi)}(L)$ for any $L \in Lit$.*

Proof. By $\mathcal{AS}(\Pi) = \mathcal{AS}(tr(\Pi))$, $\phi_\Pi(L) = x$ iff $\phi_{tr(\Pi)}(L) = x$ for $x \in \{\top, \mathbf{d}\top, \mathbf{dtc}, \mathbf{dfc}, *, \bot\}$. Since $tr(\Pi)$ is obtained from Π without **RED**$^+$, $Cn(\Pi^+) = Cn(tr(\Pi)^+)$, thereby $\phi_\Pi(L) = y$ iff $\phi_{tr(\Pi)}(L) = y$ for $y \in \{\mathbf{t}, \mathbf{f}\}$. Then, $\phi_\Pi(L) = z$ iff $\phi_{tr(\Pi)}(L) = z$ for $z \in \{\mathbf{dts}, \mathbf{dfs}\}$. Hence, the result holds. □

Corollary 3.8. *For two ELPs Π_1 and Π_2, $\Pi_1 \leqslant_{AS} \Pi_2$ iff $tr(\Pi_1) \leqslant_{AS} tr(\Pi_2)$. In particular, $\Pi_1 \simeq_{AS} \Pi_2$ iff $tr(\Pi_1) \simeq_{AS} tr(\Pi_2)$.*

The positive reduction **RED**$^+$ reduces NAF conditions and does not preserve default truth values in general.

Example 3.6. Let $\Pi = \{ p \leftarrow not\ q \}$. Applying **RED**$^+$, the program becomes $\Pi' = \{p \leftarrow\}$ where $\phi_\Pi(p) \neq \phi_{\Pi'}(p)$.

Note that **FOLD** may also reduce NAF conditions in a rule, but by its definition folding does not turn default consequences into definite ones.

Example 3.7. Let Π be the program:

$$p \leftarrow not\ q,$$
$$r \leftarrow not\ q.$$

The first rule is folded by the second rule and it becomes

$$p \leftarrow r,$$
$$r \leftarrow not\ q.$$

By the definition of **FOLD**, r appears in the head of no other rule in the program, so p is only derived by the second rule. Consequently, p remains to be default status.

4. Discussion

4.1. Multi-valued default logic

Ginsberg [13] firstly introduces a multi-valued bilattice for default logic. He distinguishes between definite and default conclusions obtained from a (super-normal) default theory using the bilattice of Fig. 2. However, Ginsberg's bilattice is seven-valued and does not distinguish between skeptical and credulous default conclusions. For instance, suppose the super-normal default theory

$$\Delta : \quad \frac{: p \wedge q}{p \wedge q}, \quad \frac{: \neg p}{\neg p},$$

which has two default extensions $Th(\{ p \wedge q\})$ and $Th(\{\neg p\})$. Then, $\phi_\Delta(p) = *$, $\phi_\Delta(q) =$ **dtc**, and $\phi_\Delta(\neg p \vee q) =$ **dts** in our framework, while Ginsberg interprets p as $*$ but handles both q and $\neg p \vee q$ as **dt**. Thus, to distinguish skeptical/credulous default inference, additional truth values are necessary as introduced in this paper. In [13], a bilattice having the same topology as X is used in the context of *prioritized default logic*, but truth values assigned to the lattice are different from ours. For super-normal default theories, Brass [6] compares different default semantics including skeptical/credulous inferences, and investigates their semantic properties. Dionísio et al. [9] distinguish skeptical/credulous default inference in super-normal default theories using modal logic. The goal of these studies is to characterize different types of default reasoning and is not ordering default theories.

From the computational viewpoint, there is a difficulty for directly computing ϕ_Δ for an arbitrary formula F. This is due to the fact that the interpretation ϕ_Δ of a formula F is generally not constructive by those of the sub-formulas of F (Proposition 2.4). The same problem happens in the restricted class of super-normal default theories [13]. For testing an order between default theories, however, Theorem 2.9 provides a sufficient condition to

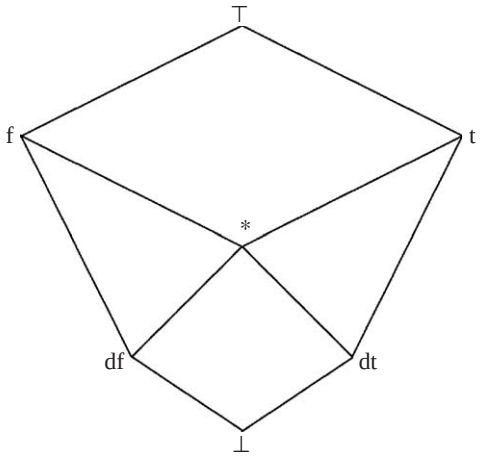

Fig. 2. Ginsberg's bilattice for default logic.

know the relation $\Delta_1 \leqslant_{DL} \Delta_2$ using default extensions. In the context of logic programming, $\Pi_1 \leqslant_{AS} \Pi_2$ is examined by Theorem 3.6 using the existing procedures for computing answer sets.

According to Gottlob [14], the complexity of checking the existence of extensions in propositional default logic is Σ_2^P-complete. And the complexity of credulous/skeptical reasoning tasks in propositional default logic is Σ_2^P/Π_2^P-complete. These results imply that deciding whether a given formula F has a truth value $x \in \{\mathbf{dtc}, \mathbf{dfc}, *, \mathbf{d}\top, \bot\}$ (resp., $x \in \{\mathbf{dts}, \mathbf{dfs}\}$) under the mapping ϕ_Δ is Σ_2^P-complete (resp., Π_2^P-complete). When $\mathcal{EXT}(\Delta) \neq \emptyset$, deciding whether a given propositional formula F has a truth value $x \in \{\mathbf{t}, \mathbf{f}, \top\}$ is the problem of propositional entailment testing in W, which is coNP-complete. On the other hand, checking the existence of answer sets in a ground ELP is NP-complete, and credulous/skeptical reasoning tasks under the answer set semantics are NP/coNP-complete [17]. This implies that the corresponding decision problems in ground ELPs are one-level lower than those in default theories. That is, deciding whether a ground literal L has a truth value $x \in \{\mathbf{dtc}, \mathbf{dfc}, *, \mathbf{d}\top, \bot\}$ (resp., $x \in \{\mathbf{dts}, \mathbf{dfs}\}$) under the mapping ϕ_Π is NP-complete (resp., coNP-complete). When $\mathcal{AS}(\Pi) \neq \emptyset$, a ground literal L has a truth value $x \in \{\mathbf{t}, \mathbf{f}, \top\}$ is decided in polynomial time.

4.2. Multi-valued semantics of logic programming

In logic programming, Fitting [11] characterizes the semantics of normal logic programs using Belnap's four-valued bilattice [4] (Fig. 3). Normal logic programs do not contain explicit negation $\neg$ in a program. The truth value **f** is then assigned to a ground atom which is a consequence of negation as failure. On the other hand, the truth value **t** is assigned to a ground atom which is either a definite conclusion of *not*-free rules or a default conclusion through negation as failure. Thus, the semantics based on the four-valued logic does not distinguish between definite and default information. Bochman [5] provides a

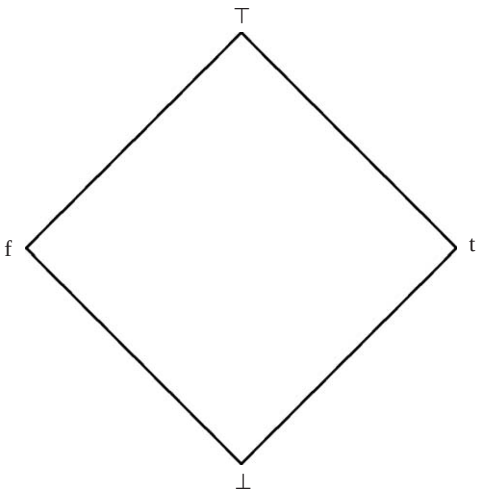

Fig. 3. Belnap's four-valued bilattice.

logical formalism called biconsequence relations for nonmonotonic reasoning. His logic characterizes Belnap's four-valued inference and semantics of logic programming, but it is unknown whether it can characterize the ten-valued default semantics. Dix [10] uses the knowledge ordering under a three-valued logic to compare information obtained from a single normal logic program under different semantics. This is in contrast to our approach in which we compare different programs under the single answer set semantics. Lattice-valued logics are also used for characterizing the "paraconsistent" semantics of logic programs [8]. Multi-valued interpretations introduced in this paper are not paraconsistent, since it is used for characterizing default logic or the answer set semantics which is not paraconsistent. However, if we use a paraconsistent version of default logic or logic programming, we can construct a paraconsistent multi-valued semantics based on the logic X. In case of the answer set semantics, this is done just by abandoning the logical closedness in its definition; we permit an answer set which is not Lit but includes both L and $\neg L$. To such literals the truth value $\top$ is assigned. For instance, the program $\{ p \leftarrow, \quad \neg p \leftarrow, \quad q \leftarrow \}$ has the paraconsistent answer set in which p has the value $\top$ and q has the value **t**. Thus, inconsistent information is localized and does not trivialize the whole program. An example of such a paraconsistent answer set semantics for extended logic programs is in [25].

Different types of multi-valued bilattices are introduced by several researchers. To our best knowledge, the ten-valued bilattice considered in this paper never appears in the literature. Moreover, existing studies all use multi-valued logics to provide a semantics of a single program, while we use them to compare information between different programs.

4.3. Order-equivalence

The order-equivalence provides a stronger relation than the normal extension-based equivalence (Proposition 2.7). Turner [27] extends the notion of *strong equivalence*

relation between logic programs [16] to default theories. Two default theories Δ_1 and Δ_2 are strongly equivalent if for every default theory Δ, $\Delta_1 \cup \Delta$ has the same extensions as $\Delta_2 \cup \Delta$.[11] For instance,

$$\Delta_1 = \left\{ \frac{:\neg q}{p} \right\} \quad \text{and} \quad \Delta_2 = \{p\}$$

are equivalent in the sense that they have the same extension $Th(\{p\})$. They are not strongly equivalent, however, as $\Delta_1 \cup \{q\}$ and $\Delta_2 \cup \{q\}$ have different extensions. Comparing the strong equivalence and the order-equivalence, there is no stronger/weaker relation between them. For instance,

$$\Delta_3 = \left\{ \frac{:\neg q}{p}, \quad \frac{q:\neg p}{p} \right\} \quad \text{and} \quad \Delta_4 = \left\{ p, \quad \frac{q:\neg p}{p} \right\}$$

are strongly equivalent but not order-equivalent. By contrast,

$$\Delta_5 = \left\{ \frac{:\neg q}{p} \right\} \quad \text{and} \quad \Delta_6 = \left\{ \frac{:\neg r}{p} \right\}$$

are order-equivalent but not strongly equivalent. The strong equivalence captures a context-independent equivalence, i.e., Δ_3 is replaced by Δ_4 in any default theory without changing the meaning of the whole theory. By contrast, the order-equivalence captures a content-based equivalence, i.e., Δ_5 and Δ_6 bring the same sort of consequences. Thus, two equivalence relations are different in their objectives and outcomes.

In logic programming, the order-equivalence is not stronger than the normal equivalence based on answer sets in general. Recall two programs in Example 3.4:

$$\Pi_1 : p \leftarrow not\ r, \quad q \leftarrow not\ r, \quad r \leftarrow not\ q,$$
$$\Pi_2 : p \leftarrow not\ q, \quad q \leftarrow not\ r, \quad r \leftarrow not\ q.$$

Π_1 and Π_2 are not equivalent as they have different answer sets $\mathcal{AS}(\Pi_1) = \{\{p, q\}, \{r\}\}$ and $\mathcal{AS}(\Pi_2) = \{\{p, r\}, \{q\}\}$. Under the interpretation ϕ_Π, however, every literal has the same truth value **dtc** in Π_1 and Π_2, which makes them order-equivalent. On the other hand, if we consider the corresponding default theories:

$$\Delta_1 : \frac{:\neg r}{p}, \quad \frac{:\neg r}{q}, \quad \frac{:\neg q}{r},$$
$$\Delta_2 : \frac{:\neg q}{p}, \quad \frac{:\neg r}{q}, \quad \frac{:\neg q}{r},$$

Δ_1 and Δ_2 have different extensions $\mathcal{EXT}(\Delta_1) = \{ Th(\{p, q\}),\ Th(\{r\}) \}$ and $\mathcal{EXT}(\Delta_2) = \{ Th(\{p, r\}),\ Th(\{q\}) \}$. In contrast to the case of logic programs, Δ_1 and Δ_2 are not order-equivalent. In fact, the formula $p \wedge q$, for instance, has different values in each theory: $\phi_{\Delta_1}(p \wedge q) =$ **dtc** and $\phi_{\Delta_2}(p \wedge q) = \bot$. Such incompatibility comes from the difference between answer sets and default extensions—the former provides the meaning of a program as a collection of literals, while the latter provides the meaning of a theory

[11] Turner defines the notion for *nested* default theories which generalize Reiter's default theories.

as a collection of formulas. As illustrated in the above example, two programs are order-equivalent as individual literals have the same interpretation under ϕ_{Π}. The corresponding two default theories are not order-equivalent as compound formulas do not necessarily have the same interpretation under ϕ_{Δ}.

4.4. Application to inductive logic programming

In the fields of machine learning and inductive logic programming, a theory of generalization has been extensively studied in the context of first-order logic [19]. However, generalization under logical entailment $\models$ is not directly applicable to default theories and nonmonotonic logic programs. This is because logical entailment represents a relation over all models of two theories, while nonmonotonic logics take some selected models into consideration. To define a generality relation over nonmonotonic theories, it is necessary to introduce an order relation apart from logical entailment. A default ordering introduced in this paper can order default theories and nonmonotonic logic programs, thereby could give a theoretical ground for inductive generalization in nonmonotonic logic programs. For instance, consider two programs:

$$\begin{aligned}\Pi_1 : \ & \mathit{flies}(x) \leftarrow \mathit{bird}(x),\ \mathit{not}\ \mathit{ab}(x),\\ & \mathit{bird}(\mathit{tweety}) \leftarrow,\\ \Pi_2 : \ & \mathit{flies}(x) \leftarrow \mathit{bird}(x),\\ & \mathit{bird}(\mathit{tweety}) \leftarrow,\end{aligned}$$

where $\phi_{\Pi_1}(\mathit{bird}(\mathit{tweety})) = \phi_{\Pi_2}(\mathit{bird}(\mathit{tweety})) = \mathbf{t}$, $\phi_{\Pi_1}(\mathit{flies}(\mathit{tweety})) = \mathbf{dts}$, $\phi_{\Pi_2}(\mathit{flies}(\mathit{tweety})) = \mathbf{t}$, and $\phi_{\Pi_1}(\mathit{ab}(\mathit{tweety})) = \phi_{\Pi_2}(\mathit{ab}(\mathit{tweety})) = \bot$. As a result, the relation $\Pi_1 \leqslant_{AS} \Pi_2$ holds. In this place, if we read the order $\leqslant_{AS}$ as "more general", Π_2 is considered a generalization of Π_1. This coincides with the view in the ILP literature [2] in which Π_1 is a specialization of Π_2. In this respect, inductive generalization of a nonmonotonic logic program Π_1 (under the answer set semantics) is considered a process of computing a program Π_2 such that $\Pi_1 \leqslant_{AS} \Pi_2$. When Π_1 and Π_2 are Horn logic programs, it holds that $\Pi_1 \leqslant_{AS} \Pi_2$ iff $\Pi_2 \models \Pi_1$ (Proposition 3.3). Thus, generalization based on $\leqslant_{AS}$ reduces to the notion of generalization under logical entailment in Horn logic programs.

Now let us look how such generalization based on $\leqslant_{AS}$ works in induction problems. First, we set the induction problem as follows: given an initial consistent program (or background knowledge) Π_0 and (positive) examples $E_1, \ldots, E_k$ as ground literals which are not skeptically entailed in Π_0, find a consistent program Π_1 which skeptically entails every example. The program Π_1 is built by adding (hypothetical) rules to Π_0. Thus, the induction problem in nonmonotonic logic programming is captured as computation of a new consistent program that skeptically entails every example. Note that induction problems often consider negative examples which should not be entailed, but here we consider positive examples only for simplicity reasons. Suppose the initial program as

$$\begin{aligned}\Pi_0 : \ & \mathit{bird}(x) \leftarrow \mathit{penguin}(x),\\ & \mathit{bird}(\mathit{tweety}) \leftarrow,\\ & \mathit{penguin}(\mathit{polly}) \leftarrow,\\ & \mathit{ostrich}(\mathit{joe}) \leftarrow .\end{aligned}$$

Given examples

$$E_1 : \mathit{flies}(\mathit{tweety}),$$
$$E_2 : \neg\mathit{flies}(\mathit{polly}),$$

suppose the following rules are induced [12]

$$H_1 : \mathit{flies}(x) \leftarrow \mathit{bird}(x),\ \mathit{not}\ \mathit{penguin}(x),$$
$$\neg\mathit{flies}(x) \leftarrow \mathit{penguin}(x).$$

The program $\Pi_1 = \Pi_0 \cup H_1$ then skeptically entails both E_1 and E_2. If another example is incrementally given as

$$E_3 : \quad \neg\mathit{flies}(\mathit{joe}),$$

the following rules are induced

$$H_2 : \mathit{flies}(x) \leftarrow \mathit{bird}(x),\ \mathit{not}\ \mathit{ab}(x),$$
$$\neg\mathit{flies}(x) \leftarrow \mathit{ab}(x),$$
$$\mathit{ab}(x) \leftarrow \mathit{penguin}(x),$$
$$\mathit{ab}(x) \leftarrow \mathit{ostrich}(x),$$

where the program $\Pi_2 = \Pi_0 \cup H_2$ entails all E_1, E_2, and E_3. Here, the relations $\Pi_0 \leqslant_{AS} \Pi_1$ and $\Pi_1 \leqslant_{AS} \Pi_2$ hold, so Π_0 is generalized to Π_1 that is generalized to Π_2. Each example having the value $\bot$ in the initial program Π_0 turns into the value **dts** or **t** in Π_1 and Π_2 after induction. Note that this naturally extends induction in Horn logic programs. In induction from Horn logic programs, given examples have the initial value $\bot$ and acquire the value **t** after induction. In induction from nonmonotonic logic programs, examples can have the value **dts** after induction. This is because induced rules are possibly default rules having negation as failure in their bodies. In this case, examples implied by those default rules are in default status.

5. Conclusion

In this paper, we have introduced multi-valued interpretations of default theories, which can distinguish between definite and skeptical/credulous default consequences. Based on this, we have developed a theory for ordering default theories, that is a natural extension of the one for (propositional) first-order theories. We have then applied the technique to ordering nonmonotonic logic programs, and shown that the order relation is preserved by most of the elementary program transformations. The notion of order-equivalence was also introduced, which is a fine-grained equivalence relation over default theories based on their information contents.

The results of this paper provide a method of comparing default theories or nonmonotonic logic programs in a manner different from the conventional extension-based or model-based

[12] This paper does not concern with detailed algorithms for induction. Examples of such algorithms are found in [15,24], for instance.

standpoint. The ten-valued bilattice introduced in this paper will be used for characterizing other nonmonotonic formalisms which have the same inference modes as default logic. The multi-valued semantics of nonmonotonic logic programs is extended to programs containing disjunctions, and different types of bilattices would be devised under different semantics. In this paper, we discussed a possible application of the proposed theory to induction from nonmonotonic logic programs. For another application, it would be used for introducing priorities over different theories and selecting information from multiple sources. An example is cooperative reasoning in multi-agent systems where an agent has to select information from conflicting agents. In this situation, theory ordering can provide a guideline for selecting information based on certainty measure. Further studies on these subjects are topics of future research.

Acknowledgements

The author thanks the anonymous referees for their valuable comments.

References

[1] C. Aravindan, P.M. Dung, On the correctness of unfold/fold transformation of normal and extended logic programs, J. Logic Programming 22 (1994) 201–217.
[2] M. Bain, S. Muggleton, Non-monotonic learning, in: S. Muggleton (Ed.), Inductive Logic Programming, Academic Press, New York, 1992, pp. 145–161.
[3] C. Baral, M. Gelfond, Logic programming and knowledge representation, J. Logic Programming 19/20 (1994) 73–148.
[4] N.D. Belnap, A useful four-valued logic, in: J.M. Dunn, G. Epstein (Eds.), Modern Uses of Multiple-Valued Logic, Reidel Publishing, Dordrecht, 1975, pp. 8–37.
[5] A. Bochman, Biconsequence relations for nonmonotonic reasoning, in: Proc. 5th Internat. Conf. on Principles of Knowledge Representation and Reasoning, Morgan Kaufmann, Los Altos, CA, 1996, pp. 482–492.
[6] S. Brass, On the semantics of supernormal defaults, in: Proc. 13th Internat. Joint Conf. Artificial Intelligence, Morgan Kaufmann, Los Altos, CA, 1993, pp. 578–583.
[7] G. Brewka, J. Dix, Knowledge representation with logic programs, in: Proc. 3rd Workshop on Logic Programming and Knowledge Representation, Lecture Notes in Artificial Intelligence, Vol. 1471, Springer, Berlin, 1997, pp. 1–51.
[8] C.V. Damásio, L.M. Pereira, A survey of paraconsistent semantics for logic programs, in: D.M. Gabbay, Ph. Smets (Eds.), Handbook of Defeasible Reasoning and Uncertainty Management Systems, Vol. 2, Kluwer Academic, Dordrecht, 1998, pp. 241–320.
[9] F.M. Dionísio, S. Brass, M. Ryan, U. Lipeck, Hypothetical reasoning with defaults, in: Proc. Workshop on Computational Aspects of Nonmonotonic Reasoning, at the 7th Internat. Workshop on Nonmonotonic Reasoning, 1998.
[10] J. Dix, A framework for representing and characterizing semantics of logic programs, in: Proc. 3rd Internat. Conf. on Knowledge Representation and Reasoning, Morgan Kaufmann, Los Altos, CA, 1992, pp. 591–602.
[11] M. Fitting, Bilattices and the semantics of logic programming, J. Logic Programming 11 (1991) 91–116.
[12] M. Gelfond, V. Lifschitz, Classical negation in logic programs and disjunctive databases, New Generation Comput. 9 (1991) 365–385.
[13] M.L. Ginsberg, Multi-valued logics: a uniform approach to inference in artificial intelligence, Comput. Intell. 4 (1988) 265–316.
[14] G. Gottlob, Complexity results for nonmonotonic logics, J. Logic Comput. 2 (1992) 397–425.
[15] K. Inoue, Y. Kudoh, Learning extended logic programs, in: Proc. 15th Internat. Joint Conf. on Artificial Intelligence, Morgan Kaufmann, Los Altos, CA, 1997, pp. 176–181.

[16] V. Lifschitz, D. Pearce, A. Valverde, Strongly equivalent logic programs, ACM Trans. Comput. Logic 2 (2001) 526–541.
[17] W. Marek, M. Truszczynski, Autoepistemic logic, J. Assoc. Comput. Math. 38 (3) (1991) 588–619.
[18] T. Niblett, A study of generalization in logic programs, in: Proc. 3rd European Working Sessions on Learning (EWSL-88), Pitman, London, 1988, pp. 131–138.
[19] S.-H. Nienhuys-Cheng, R. de Wolf, Foundations of Inductive Logic Programming, Lecture Notes in Artificial Intelligence, Vol. 1228, Springer, Berlin, 1997.
[20] R. Reiter, A logic for default reasoning, Artificial Intelligence 13 (1980) 81–132.
[21] S. Russell, P. Norvig, Artificial Intelligence: A Modern Approach, second ed., Prentice-Hall, Englewood Cliffs, NJ, 1995.
[22] C. Sakama, Nonmonotonic inductive logic programming, in: Proc. 6th Internat. Conf. on Logic Programming and Nonmonotonic Reasoning, Lecture Notes in Artificial Intelligence, Vol. 2173, Springer, Berlin, 2001, pp. 62–80.
[23] C. Sakama, Ordering default theories, in: Proc. 18th Internat. Joint Conf. on Artificial Intelligence, Morgan Kaufmann, Los Altos, CA, 2003, pp. 839–844.
[24] C. Sakama, Induction from answer sets in nonmonotonic logic programs, ACM Trans. Comput. Logic 6 (2) (2005).
[25] C. Sakama, K. Inoue, Paraconsistent stable semantics for extended disjunctive programs, J. Automat. Reason. 13 (1) (1995) 145–172.
[26] H. Tamaki, T. Sato, Unfold/fold transformation of logic programs, in: Proc. 2nd Internat. Conf. on Logic Programming, 1984, pp. 127–138.
[27] H. Turner, Strong equivalence for logic programs and default theories, (made easy), in: Proc. 6th Internat. Conf. on Logic Programming and Nonmonotonic Reasoning, Lecture Notes in Artificial Intelligence, Vol. 2173, Springer, Berlin, 2001, pp. 81–92.