Noname manuscript No.
(will be inserted by the editor)

Inductive Equivalence in Clausal Logic and Nonmonotonic Logic Programming

Chiaki Sakama · Katsumi Inoue

Received: date / Accepted: date

Abstract This paper provides a logical framework for comparing inductive capabilities among agents having different background theories. A background theory is called *inductively equivalent* to another background theory if two theories induce the same hypotheses for any observation. Conditions of inductive equivalence change depending on the logic of representation languages and the logic of induction or inductive logic programming (ILP). In this paper, we consider clausal logic and nonmonotonic logic programs as representation languages for background theories. Then, we investigate conditions of inductive equivalence in four different frameworks of induction, *cautious induction*, *brave induction*, *learning from satisfiability*, and *descriptive induction*. We observe that several induction algorithms in Horn ILP systems require weaker conditions of equivalence under restricted problem settings. We address that inductive equivalence can be used for verification and evaluation of induction algorithms, and argue problems for optimizing background theories in ILP.

Keywords inductive equivalence · inductive logic programming · nonmonotonic logic programs

1 Introduction

Equivalence relations between logical theories have been studied in many ways in artificial intelligence and logic programming. In knowledge representation, a theory represents knowledge of a problem domain. The same problem would be represented in different ways by different experts. Equivalence of two theories is then used for evaluating information contents and identifying different information sources. In program

C. Sakama
Department of Computer and Communication Sciences
Wakayama University, Sakaedani, Wakayama 640-8510 Japan
Phone/Fax: +81-73-457-8128,
E-mail: sakama@sys.wakayama-u.ac.jp

K. Inoue
National Institute of Informatics
2-1-2 Hitotsubashi, Chiyoda-ku, Tokyo, 101-8430 Japan
E-mail: ki@nii.ac.jp

development, one program may give a declarative specification of some problem and another program may give an efficient coding of it. In this case, equivalence of two programs guarantees a correct implementation of the given specification. In the context of logic programming, various criteria for equivalence relations are proposed in the literature (Maher 1988; Sagiv 1988; Lifschitz et al. 2001; Eiter and Fink 2003; Inoue and Sakama 2004). Among them, weak equivalence and strong equivalence of two programs are particularly important. Two logic programs P_1 and P_2 are *(weakly) equivalent* if they have the same declarative meaning. On the other hand, two programs P_1 and P_2 are *strongly equivalent* if they preserve the equivalence relation by the introduction of arbitrary rules R to them.

Equivalence relations presented above compare capabilities of deductive reasoning between programs. For instance, two Horn logic programs are weakly equivalent if they have the same least model that is the logical consequences of each program. When we consider realizing intelligent agents that can perform commonsense reasoning, however, comparing capabilities of *non-deductive* reasoning between programs is also necessary and important. Recently, Inoue and Sakama (2005, 2006a,b) argue equivalence in *abductive logic*. They introduce two different types of abductive equivalence: *explainable equivalence* and *explanatory equivalence*. The former considers whether two theories have the same explainability for any observation, while the latter considers whether two theories have the same explanation contents for any observation. These two notions compare capabilities of abductive reasoning among agents, and they provide necessary and sufficient conditions for abductive equivalence in first-order logic and *abductive logic programming* (ALP) (Denecker and Kakas 2002). *Induction* is also known as non-deductive reasoning, which is often distinguished from abduction (Flach and Kakas 2000). In computational logic, induction is realized by *inductive logic programming* (ILP) (Muggleton 1992; Nienhuys-Cheng and Wolf 1997). A typical induction problem is to build a hypothesis which covers a given observation with respect to a background theory. Then, there are some questions concerning equivalence issues in induction.

1. *When can we say that induction with a background theory is equivalent to induction with another background theory?*
   Two different background theories B_1 and B_2 are considered equivalent if they induce the same hypothesis H for any observation O. This equivalence measure is useful for comparing "information contents" of different background theories.
2. *When can we say that induced hypotheses are equivalent to another induced hypotheses?*
   Two hypotheses H_1 and H_2 are considered equivalent if they account for the same observation O with respect to a background theory B. This equivalence measure is useful for comparing "explanation power" of different hypotheses.
3. *When can we say that induction from an observation is equivalent to induction from another observation?*
   Two observations O_1 and O_2 are considered equivalent if they produce the same hypothesis H with respect to a background theory B. This equivalence measure is useful for comparing "evidential power" of different observations.
4. *Do conditions for these equivalence differ by underlying logics?*
   The results of induction and equivalence conditions generally depend on a logic on which induction is based. Moreover, those conditions differ among individual induction algorithms. Then, we can compare different logics or algorithms for estimating their induction capabilities.

These issues are important and meaningful for comparing different induction tasks, but few studies have argued the problems so far. In this paper, we focus on the question (1) above and study the problem of equivalence of background theories in induction. To answer the question (4), we also investigate conditions of equivalence in different logics and induction algorithms. Other problems, concerning the questions (2) and (3), are studied in a different paper (Sakama and Inoue 2009a).

To formalize the problem, we introduce the notion of *inductive equivalence* between background theories. A background theory B_1 is said *inductively equivalent* to another background theory B_2 if B_1 and B_2 induce the same hypothesis H for an arbitrary observation O. Intuitively, if an agent has a background theory B_1 which is inductively equivalent to another background theory B_2 of another agent, then these two agents are considered equivalent with respect to inductive capability. In this case, we can identify those two agents as far as induction is concerned. From the viewpoint of program development, if a theory B_1 is transformed to another syntactically different B_2, inductive equivalence of two theories guarantees identification of results of induction from each theory. This provides guidelines for optimizing background theories in ILP. The problem of interest is logical conditions for inductive equivalence in ILP. Conditions for inductive equivalence differ depending on logics of representation languages and logics of induction. This paper considers two logics for representation languages – *clausal logic* and *nonmonotonic logic programming*. These logics are widely used in knowledge representation and ILP (Muggleton 1992; Baral and Gelfond 1994). On the other hand, we consider four different frameworks of induction, *cautious induction*, *brave induction* (Sakama and Inoue 2009b), *learning from satisfiability* (De Raedt 1997; De Raedt and Dehaspe 1997a), and *descriptive induction* (Lachiche 2000). These frameworks capture different aspects of induction problems. We show necessary and sufficient conditions for inductive equivalence under different semantics in respective induction. We also observe that some induction algorithms in Horn ILP systems require weaker conditions of inductive equivalence. We address that inductive equivalence is used for testing correctness/completeness of an induction algorithm and comparing capabilities of different algorithms. We also argue problems for optimizing background theories in ILP through appropriate program transformations.

This paper is a revised and extended version of (Sakama and Inoue 2005). The differences between the present work and the previous one are follows. First, we apply the framework of inductive equivalence to different types of induction, and investigate formal properties among them. Inductive equivalences in cautious induction, brave induction, and learning from satisfiability are new in this paper. Second, inductive equivalences in particular induction algorithms are also revised and extended. Inductive equivalences in FOIL and BRAIN^{not} are new in this paper. Third, previous results for inductive equivalence in nonmonotonic logic programs are generalized to background theories which possibly contain disjunctions. Fourth, new considerations and additional arguments are added throughout the paper.

The rest of this paper is organized as follows. Section 2 presents logical frameworks used in this paper. Section 3 introduces the notion of inductive equivalence and investigates formal properties in clausal logic. Section 4 verifies conditions of inductive equivalence in some Horn ILP systems. Section 5 applies inductive equivalence to nonmonotonic logic programs. Section 6 discusses related issues and potential applications. Finally, Section 7 concludes the paper.

2 Logical Framework

2.1 Clausal Theories

A *first-order language* consists of an alphabet and all formulas defined over it. The definition is the standard one in the literature (Nienhuys-Cheng and Wolf (1997), for instance). A *first-order theory* is a set of formulas. A *clause* is a formula of the form

$$A_1 \vee \cdots \vee A_m \vee \neg A_{m+1} \vee \cdots \vee \neg A_n \tag{1}$$

where A_i $(1 \leq i \leq n)$ are atoms and every variable appearing in (1) is universally quantified at the front. A *clausal theory* (or simply a *theory*) is a set of clauses. The clause (1) is also written as

$$A_1 \vee \cdots \vee A_m \leftarrow A_{m+1}, \ldots, A_n \tag{2}$$

in the context of *logic programming*. The disjunction $A_1 \vee \cdots \vee A_m$ is the *head* and the conjunction $A_{m+1}, \ldots, A_n$ is the *body* of the clause. In particular, a clause (2) having at most one atom in its head is a *Horn clause* and a set of Horn clauses is a *Horn logic program*. A Horn clause is called a *definite clause* if it contains exactly one atom in its head. A *definite logic program* is a set of definite clauses. A clause $A \leftarrow$ is called a *fact* and is identified with the atom A. A theory is identified with the conjunction of clauses included in the theory. A theory, a clause or an atom is *ground* if it contains no variable. A theory or a clause with variables is identified with the set of its ground instances. A *propositional theory* is a finite set of ground clauses. Clausal theories and Horn logic programs are subsets of first-order theories, while nonmonotonic logic programs, which are handled in Section 4, are outside of first-order logic.

The domain of a theory is given as the *Herbrand universe* and interpretations are defined as subsets of the *Herbrand base* HB. An interpretation M *satisfies* the ground clause of the form (2) if $\{ A_{m+1}, \ldots, A_n \} \subseteq I$ implies $\{ A_1, \ldots, A_m \} \cap I \neq \emptyset$. M *satisfies* a theory T if M satisfies every ground instance of any clause in T. An interpretation M is a *model* of a theory T if M satisfies T. The set of all models of T is written as $Mod(T)$. The *semantics* of a theory T is represented as a subset $SEM(T)$ of $Mod(T)$, i.e.,

$$SEM(T) \subseteq Mod(T).$$

The set $SEM(T)$ represents models that are selected from $Mod(T)$ based on some preference criterion. In particular, $SEM(T) = Mod(T)$ holds under the classical model theory of first-order logic. A theory T is *consistent* under SEM if $SEM(T) \neq \emptyset$; otherwise, T is *inconsistent*. A theory T *satisfies* a clause C (written as $T \models C$) if C is satisfied in every model of T. T *satisfies* a set S of clauses (written as $T \models S$) if $T \models C$ for any clause C in S. There are several criteria for selecting models as $SEM(T)$. Among them, *minimal models* are often considered in the literature. The set of all minimal models of T (denoted by $MM(T)$) is defined as

$$MM(T) = \{ M \in Mod(T) \mid \neg \exists N \in Mod(T) \text{ such that } N \subset M \}.$$

Every consistent clausal theory has a minimal model (Bossu and Siegel 1985).

2.2 Logics of Induction

There are several definitions of induction. In this paper, we consider the following four different frameworks of induction:

- *Cautious induction* (Sakama and Inoue 2009b)
- *Brave induction* (Sakama and Inoue 2009b)
- *Learning from satisfiability* (De Raedt 1997; De Raedt and Dehaspe 1997a)
- *Descriptive induction* (Lachiche 2000)

Let B, H, and O be sets of formulas respectively representing a *background theory*, a *hypothesis*, and an *observation*. Then, each induction is defined as follows.[1]

Cautious induction: Given B and O, find H such that O is satisfied by every $M \in SEM(B \cup H)$ where $B \cup H$ is consistent.
Brave induction: Given B and O, find H such that O is satisfied by some $M \in SEM(B \cup H)$ where $B \cup H$ is consistent.
Learning from satisfiability: Given B and O, find H such that $B \cup H \cup O$ is consistent under SEM.
Descriptive induction: Give B and O, find H such that H is satisfied by every $M \in SEM(B \cup O)$ where $B \cup O$ is consistent.

In each case, we say that a hypothesis H *covers* (or *explains*) O with respect to B (under SEM) in the induction framework I. H is also called a *solution* in I. Here, I is one of the four induction frameworks presented above. In this paper, cautious induction, brave induction, learning from satisfiability, and descriptive induction is respectively abbreviated as CauInd, BraInd, LFS, and DesInd.

Cautious induction requires an observation to be satisfied in *every* model in $SEM(B \cup H)$. In particular, when $SEM(B \cup H) = Mod(B \cup H)$ in first-order logic, it is written as

$$B \cup H \models E.$$

In this case, cautious induction is also called *explanatory induction* (abbreviated as ExpInd) (Flach 1996), which is known as usual setting in ILP (Muggleton 1992; Nienhuys-Cheng and Wolf 1997). Brave induction, on the other hand, requires that an observation is satisfied in *some* models in $SEM(B \cup H)$. By the definition, brave induction is weaker than cautious induction, that is, if H is a solution of cautious induction, it is also a solution of brave induction, but not vice versa. Learning from satisfiability is weaker than brave induction, so that it provides the weakest form of induction among those three frameworks. Brave induction and learning from satisfiability coincide when $SEM(B) = Mod(B)$. That is, O is satisfied in some model of $B \cup H$ iff $B \cup H \cup O$ is consistent. Descriptive induction, which is also called *confirmatory induction* (Flach 1996), prescribes that a hypothesis is satisfied in a background theory and an observation. In contrast to explanatory induction, it does not intend to learn classification rules but seek regularities over observed data. When an observation is given as a set of interpretations, descriptive induction is also called *learning from interpretations* (De Raedt 1997; De Raedt and Dehaspe 1997b).

[1] Observations defined here are *positive observations*. In the literature, *negative* observations are often considered as well as positive ones. For simplicity reasons, we consider positive observations only in this paper.

Example 2.1 (Sakama and Inoue 2009b) Suppose that there are 30 students in a class, of which 20 are European, 7 are Asian, and 3 are American. The situation is represented by the background theory B and the observation O:

$$B = \{\ student(1), \ldots, student(30)\ \},$$
$$O = \{\ euro(1), \ldots, euro(20),\ asia(21), \ldots, asia(27),\ usa(28), \ldots, usa(30)\ \}$$

where each number represents individual students. Put the semantics of the background theory as the minimal model semantics, $SEM(B) = MM(B)$. First, consider the set H_1 of the following clauses

$$euro(x) \leftarrow student(x),$$
$$asia(x) \leftarrow student(x),$$
$$usa(x) \leftarrow student(x).$$

Then, H_1 is a solution of BraInd, CauInd, and LFS, but it is not a solution of DesInd.

Next, consider the set H_2 which consists of the single clause

$$euro(x) \vee asia(x) \vee usa(x) \ \leftarrow\ student(x)\,.$$

H_2 is a solution of BraInd, LFS, and DesInd, but it is not a solution of CauInd.

Finally, consider the set H_3 which consists of the single clause

$$student(x) \ \leftarrow\ euro(x)\,.$$

H_3 is a solution of DesInd and LFS, but it is not a solution of BraInd nor CauInd.

Thus, four induction frameworks provide different solutions in general.

2.3 Equivalence Relation

Two different theories are equivalent in many ways. In this paper, we handle three different notions of equivalences. Consider two theories T_1 and T_2 which have the common underlying language. Then, T_1 and T_2 are

- *logically equivalent* (written as $T_1 \equiv T_2$) if $Mod(T_1) = Mod(T_2)$.
- *weakly equivalent* (written as $T_1 \equiv_w T_2$) if $SEM(T_1) = SEM(T_2)$.
- *strongly equivalent* (written as $T_1 \equiv_s T_2$) if $T_1 \cup U \equiv_w T_2 \cup U$ for any theory U under the same language.

By the definition, $T_1 \equiv_s T_2$ implies $T_1 \equiv_w T_2$. In particular, three equivalence relations coincide in first-order logic under the condition $SEM(T) = Mod(T)$ (Eiter and Fink 2003).

We first show that logical equivalence coincides with strong equivalence when $SEM(T) = MM(T)$ in clausal logic.

Proposition 2.1 *Let T be a clausal theory and HB its Herbrand base. For any $M (\subseteq HB)$, put $M^* = M \cup \{\ \neg A \mid A \in HB \setminus M\ \}$. Then, M is a model of a theory T iff $T \cup M^*$ is consistent.*

Proof If M is a model of T, so is M^*. Then, $T \cup M^*$ is consistent. Conversely, when $T \cup M^*$ is consistent, assume that M is not a model of T. Then, there is a clause C in T which is not satisfied by M. In this case, $M^* \cup \{C\}$ is inconsistent. This contradicts the fact that $T \cup M^*$ is consistent. □

Proposition 2.2 *(logical equivalence vs. strong equivalence under MM) Let T_1 and T_2 be two clausal theories. Then, $T_1 \equiv T_2$ iff $MM(T_1 \cup U) = MM(T_2 \cup U)$ for any clausal theory U.*

Proof The only-if part is obvious. Assume $MM(T_1 \cup U) = MM(T_2 \cup U)$ for any U. If $T_1 \not\equiv T_2$, there is either $M \in Mod(T_1) \setminus Mod(T_2)$ or $M \in Mod(T_2) \setminus Mod(T_1)$. Consider the case $M \in Mod(T_1) \setminus Mod(T_2)$. Since M is not a model of T_2, $T_2 \cup M^*$ is inconsistent (Proposition 2.1). Thus, $MM(T_2 \cup M^*) = \emptyset$. On the other hand, $M \in Mod(T_1)$ implies $M \in Mod(T_1 \cup M^*)$. Since $T_1 \cup M^*$ is a clausal theory, $M \in Mod(T_1 \cup M^*)$ implies the existence of minimal models. Hence, $MM(T_1 \cup M^*) \neq \emptyset$. This contradicts the assumption. The case of $M \in Mod(T_2) \setminus Mod(T_1)$ is proved in the same manner. Hence, $T_1 \equiv T_2$. □

Proposition 2.3 *(logical equivalence vs. weak equivalence under MM) Let T_1 and T_2 be two clausal theories. Then, $T_1 \equiv T_2$ implies $T_1 \equiv_w T_2$ under the minimal model semantics.*

The converse of Proposition 2.3 does not hold in general.

Example 2.2 Consider three clausal theories:

$$T_1 = \{\, a \vee b, \quad c \vee \neg a, \quad c \vee \neg b \,\},$$
$$T_2 = \{\, a \vee b, \quad c \,\},$$
$$T_3 = \{\, a \vee b, \quad \neg a \vee \neg b, \quad c \,\}.$$

First, set $SEM(T_i) = Mod(T_i)$ for $i = 1, 2, 3$. Then,

$$Mod(T_1) = Mod(T_2) = \{\{a, c\}, \{b, c\}, \{a, b, c\}\} \text{ and } Mod(T_3) = \{\{a, c\}, \{b, c\}\}.$$

In this case, the following relations hold: $T_1 \equiv T_2$, $T_1 \not\equiv T_3$ and $T_2 \not\equiv T_3$.

Next, set $SEM(T_i) = MM(T_i)$ for $i = 1, 2, 3$. Then,

$$MM(T_1) = MM(T_2) = MM(T_3) = \{\{a, c\}, \{b, c\}\}.$$

In this case, the following relations hold: $T_1 \equiv_w T_2 \equiv_w T_3$, $T_1 \equiv_s T_2$, $T_1 \not\equiv_s T_3$ and $T_2 \not\equiv_s T_3$. Here, $T_2 \not\equiv_s T_3$ because the addition of $Q = \{a, b\}$ makes T_3 inconsistent.

3 Inductive Equivalence in Clausal Logic

3.1 Inductive Equivalence

We first provide a general framework of inductive equivalence between two theories.

Definition 3.1 (inductive equivalence) Let B_1 and B_2 be two background theories having the same Herbrand base HB. For any observation O, suppose that a hypothesis H covers O with respect to B_1 under SEM in induction I iff H covers O with respect to B_2 under SEM in induction I. In this case, B_1 and B_2 are said to be *inductively equivalent* under SEM in I (written $B_1 \equiv_I^{SEM} B_2$).

By the definition, inductive equivalence presents that two background theories have the same explanation contents for any observation. Note that there are at least three different parameters on which inductive equivalence depends – (i) syntax of B, H and O, (ii) the underlying semantics SEM, (iii) and the framework of induction I. In this paper, we study several cases of inductive equivalence with different parameters.

The notion of inductive equivalence is applied to four induction frameworks as follows.

Definition 3.2 (inductive equivalence in different frameworks of induction) Let B_1 and B_2 be two theories having the same Herbrand base HB. Then,

1. B_1 and B_2 are *inductively equivalent under SEM in cautious induction* (written $B_1 \equiv^{SEM}_{\mathsf{CauInd}} B_2$) if for any O and any H, O is satisfied by every $M \in SEM(B_1 \cup H)$ iff O is satisfied by every $M \in SEM(B_2 \cup H)$, where $B_1 \cup H$ and $B_2 \cup H$ are consistent.
2. B_1 and B_2 are *inductively equivalent under SEM in brave induction* (written $B_1 \equiv^{SEM}_{\mathsf{BraInd}} B_2$) if for any O and any H, O is satisfied by some $M \in SEM(B_1 \cup H)$ iff O is satisfied by some $M \in SEM(B_2 \cup H)$, where $B_1 \cup H$ and $B_2 \cup H$ are consistent.
3. B_1 and B_2 are *inductively equivalent under SEM in learning from satisfiability* (written $B_1 \equiv^{SEM}_{\mathsf{LFS}} B_2$) if for any O and any H, $B_1 \cup H \cup O$ is consistent iff $B_2 \cup H \cup O$ is consistent.
4. B_1 and B_2 are *inductively equivalent under SEM in descriptive induction* (written $B_1 \equiv^{SEM}_{\mathsf{DesInd}} B_2$) if for any O and any H, H is satisfied by every $M \in SEM(B_1 \cup O)$ iff H is satisfied by every $M \in SEM(B_2 \cup O)$, where $B_1 \cup O$ and $B_2 \cup O$ are consistent.

Proposition 3.1 *(relations between different inductive equivalences) For any SEM, the following relations hold.*

1. $B_1 \equiv^{SEM}_{\mathsf{CauInd}} B_2$ *implies* $B_1 \equiv^{SEM}_{\mathsf{BraInd}} B_2$.
2. $B_1 \equiv^{SEM}_{\mathsf{BraInd}} B_2$ *implies* $B_1 \equiv^{SEM}_{\mathsf{LFS}} B_2$.
3. $B_1 \equiv^{SEM}_{\mathsf{CauInd}} B_2$ *iff* $B_1 \equiv^{SEM}_{\mathsf{DesInd}} B_2$.

Proof The results of (1) and (2) hold by their definition. To see (3), the definition of inductive equivalence in descriptive induction is obtained by exchanging the positions of O and H in the definition in cautious induction. Since both O and H are arbitrary sets of formulas, the result holds. □

Proposition 3.1(1) and (2) show that the implication relations among three induction frameworks are inherited to equivalence relations. On the other hand, Proposition 3.1(3) represents that in the context of inductive equivalence, distinction between cautious induction and descriptive induction is unimportant. With this reason, we mainly consider inductive equivalence in cautious induction, brave induction, and learning from satisfiability, hereafter.

3.2 Inductive Equivalence between Clausal Theories

In this section, we consider the following problem setting:

- a background theory B is given as a clausal theory,
- an observation O is a set of clauses,
- a hypothesis H is a set of clauses.

We first set $SEM(B) = Mod(B)$, i.e, the classical semantics in first-order logic. In this case, cautious induction coincides with explanatory induction, and brave induction coincides with learning from satisfiability, as presented in Section 2.2. Necessary and sufficient conditions for inductive equivalence are stated below.

Theorem 3.2 *(condition for inductive equivalence in explanatory induction) For any two clausal theories B_1 and B_2, $B_1 \equiv^{Mod}_{\mathsf{CaulInd}} B_2$ iff $B_1 \equiv B_2$.*

Proof Let O and H be arbitrary sets of clauses. Then, B_1 and B_2 are inductively equivalent in cautious induction
iff $B_1 \cup H \models O \Leftrightarrow B_2 \cup H \models O$ for any O and H such that $B_1 \cup H$ and $B_2 \cup H$ are consistent
iff $B_1 \models H \rightarrow O \Leftrightarrow B_2 \models H \rightarrow O$ for any O and H such that $B_1 \cup H$ and $B_2 \cup H$ are consistent
iff $B_1 \equiv B_2$. □

Theorem 3.3 *(condition for inductive equivalence in brave induction) For any two clausal theories B_1 and B_2, $B_1 \equiv^{Mod}_{\mathsf{BraInd}} B_2$ iff $B_1 \equiv B_2$.*

Proof Let O and H be arbitrary sets of clauses. Then, B_1 and B_2 are inductively equivalent in brave induction
iff $B_1 \cup H \cup O$ is consistent $\Leftrightarrow B_1 \cup H \cup O$ is consistent for any O and H
iff $B_1 \cup O$ is consistent $\Leftrightarrow B_1 \cup O$ is consistent for any O
iff $B_1 \not\models \neg O \Leftrightarrow B_2 \not\models \neg O$ for any O
iff $B_1 \models F \Leftrightarrow B_2 \models F$ for any formula F
iff $B_1 \equiv B_2$. □

By Theorems 3.2 and 3.3, the following result follows.

Corollary 3.4 *(inductive equivalence in cautious induction and brave induction) For any two clausal theories B_1 and B_2, $B_1 \equiv^{Mod}_{\mathsf{CaulInd}} B_2$ iff $B_1 \equiv^{Mod}_{\mathsf{BraInd}} B_2$.*

By the fact that descriptive induction is identified with cautious induction (Proposition 3.1), we conclude that the inductive equivalence relations in four different induction frameworks coincide under the classical semantics.

Next, we set $SEM(B) = MM(B)$ for the semantics of a clausal theory B. This setting is considered as the *minimal model semantics* of disjunctive logic programs (Minker 1982) or *circumscription* (McCarthy 1980). In this case, we have the next result.

Theorem 3.5 *(condition for inductive equivalence under MM in cautious induction) For any two clausal theories B_1 and B_2, $B_1 \equiv^{MM}_{\mathsf{CaulInd}} B_2$ iff $B_1 \equiv B_2$.*

Proof Suppose that B_1 and B_2 are inductively equivalent under the minimal model semantics in CaulInd. Then, for any set O and for any set H of clauses, O is satisfied by any $M \in MM(B_1 \cup H)$ iff O is satisfied by any $N \in MM(B_2 \cup H)$ where $B_1 \cup H$ and $B_2 \cup H$ are consistent. By putting $O = B_1 \cup H$, it holds that $B_1 \cup H$ is satisfied by any $M \in MM(B_1 \cup H)$ iff $B_1 \cup H$ is satisfied by any $N \in MM(B_2 \cup H)$. By putting

$O = B_2 \cup H$, it holds that $B_2 \cup H$ is satisfied by any $M \in MM(B_1 \cup H)$ iff $B_2 \cup H$ is satisfied by any $N \in MM(B_2 \cup H)$. As $B_1 \cup H$ is satisfied by any $M \in MM(B_1 \cup H)$ and $B_2 \cup H$ is satisfied by any $N \in MM(B_2 \cup H)$, it holds that $B_1 \cup H$ is satisfied by any $N \in MM(B_2 \cup H)$ and $B_2 \cup H$ is satisfied by any $M \in MM(B_1 \cup H)$. Since any minimal model M of $B_1 \cup H$ satisfies every clause in $B_2 \cup H$, $M \in Mod(B_2 \cup H)$. If $M \notin MM(B_2 \cup H)$, there is a minimal model $I \in MM(B_2 \cup H)$ such that $I \subset M$ and I satisfies $B_2 \cup H$. Since any minimal model I of $B_2 \cup H$ satisfies every clause in $B_1 \cup H$, $I \in Mod(B_1 \cup H)$. But this is impossible because M is a minimal model of $B_1 \cup H$. Hence, $M \in MM(B_2 \cup H)$. Likewise, $N \in MM(B_2 \cup H)$ implies $N \in MM(B_1 \cup H)$. Therefore, $MM(B_1 \cup H) = MM(B_2 \cup H)$, so that $B_1 \equiv B_2$ by Proposition 2.2.

Conversely, if $B_1 \equiv B_2$, $MM(B_1 \cup H) = MM(B_2 \cup H)$ holds for any set H of clauses (Proposition 2.2). Then, for any set H and for any set O of clauses, O is satisfied by any $M \in MM(B_1 \cup H)$ iff O is satisfied by any $N \in MM(B_2 \cup H)$, and $B_1 \cup H$ is consistent iff $B_2 \cup H$ is consistent. Hence, B_1 and B_2 are inductively equivalent under the minimal model semantics in CaInd. □

Theorem 3.6 *(condition for inductive equivalence under MM in learning from satisfiability) For any two clausal theories B_1 and B_2, $B_1 \equiv^{MM}_{\mathsf{LFS}} B_2$ iff $B_1 \equiv B_2$.*

Proof Suppose that B_1 and B_2 are inductively equivalent under the minimal model semantics in LFS. Then, for any set H and for any set O of clauses, $B_1 \cup H \cup O$ is consistent iff $B_2 \cup H \cup O$ is consistent. This condition reduces to that $B_1 \cup H$ is consistent iff $B_2 \cup H$ is consistent for any set H of clauses $(*)$. If $B_1 \not\equiv B_2$, there is a ground clause C such that $B_1 \models C$ but $B_2 \not\models C$. Then, $B_1 \cup \{\neg C\}$ is inconsistent, while $B_2 \cup \{\neg C\}$ is consistent. As $\neg C$ is a conjunction of ground literals and is identified with a set of clauses, this contradicts the fact $(*)$. Thus, the condition implies the equivalence relation $B_1 \equiv B_2$. The converse implication clearly holds. □

Theorem 3.7 *(condition for inductive equivalence under MM in brave induction) For any two clausal theories B_1 and B_2, $B_1 \equiv^{MM}_{\mathsf{BraInd}} B_2$ iff $B_1 \equiv B_2$.*

Proof As inductive equivalence in CaInd implies inductive equivalence in BraInd (Proposition 3.1), B_1 and B_2 are inductively equivalent under the minimal model semantics in BraInd if $B_1 \equiv B_2$ by Theorem 3.5. On the other hand, inductive equivalence in BraInd implies inductive equivalence in LFS (Proposition 3.1), so B_1 and B_2 are inductively equivalent under the minimal model semantics in BraInd only if $B_1 \equiv B_2$ by Theorem 3.6. □

By the results of Theorems 3.5, 3.6 and 3.7, together with those results under the classical semantics, we conclude that

Theorem 3.8 *(identification of inductive equivalence in clausal theories) For any two clausal theories B_1 and B_2, $B_1 \equiv^{SEM}_{I} B_2$ iff $B_1 \equiv B_2$ where SEM is either Mod or MM and $I \in \{\mathsf{CaInd}, \mathsf{BraInd}, \mathsf{LFS}, \mathsf{DesInd}\}$.*

Deciding the inductive equivalence of two theories is intractable in general.[2]

Proposition 3.9 *(complexity for deciding inductive equivalence between clausal theories) Deciding inductive equivalence of two propositional clausal theories is coNP-complete under both the classical semantics and the minimal model semantics in four*

[2] Throughout the paper, complexity results are stated in terms of the size of input background theories.

different induction frameworks. The task is done in polynomial time when two theories are Horn.

Proof Given two propositional clausal theories B_1 and B_2, the problem of testing $B_1 \equiv B_2$ is equivalent to the problem of testing unsatisfiability of $(B_1 \wedge \neg B_2) \vee (\neg B_1 \wedge B_2)$, which is coNP-complete. Then, the result follows by Theorem 3.8. Next, given two Horn logic programs B_1 and B_2, $B_1 \supset B_2$ is checked by testing unsatisfiability of $B_1 \cup \{\neg\, c\}$ for each clause $c \in B_2$. As Horn SAT is linear, this is done in quadratic time. The converse implication is checked in the same manner. □

4 Inductive Equivalence in Horn ILP Systems

There are many ILP systems which use Horn logic programs as background theories. In these systems, the condition of inductive equivalence is often relaxed. In this section, we investigate inductive equivalence in some Horn ILP systems which are widely studied in the literature.

4.1 Foil

Foil (Quinlan 1990) induces function-free definite clauses which cover a positive observation and uncover a negative observation together with a background theory. Given a predicate p to be learned, it starts with the fact $p(x_1, ..., x_n) \leftarrow$ which is then specialized using *refinement operators* that adds new literals to the body of the clause. Foil repeatedly applies a refinement operator until the clause does not imply any fact included in a negative observation for the predicate. Once a clause is added to a hypothesis, every ground fact implied by that clause is deleted from a positive observation. The algorithm repeats the step until all facts in a positive observation are covered. Foil uses an information-based heuristic to guide its search for hypotheses.

The logic for induction in Foil is explanatory induction with a restricted problem setting. Given a function-free definite logic program B (called a *Datalog*) and a set O of ground facts, a hypothesis H covers O with respect to B in Foil if

$$B \cup H \models O \tag{3}$$

where H is a set of function-free definite clauses satisfying the condition:[3]

$$H = \{\, C \mid \text{the predicate appearing in the head of a clause } C \text{ appears in } O \,\}.$$

The declarative semantics of a definite logic program B is given by the unique minimal model M_B, called the *least model*. The least model has the *model intersection property* (van. Emden and Kowalski 1976) such that

$$M_B = \bigcap_{M \in Mod(B)} M\,.$$

[3] In (Quinlan 1990), H contains *non-Horn* clauses having negative literals in its body, but the author explains Foil as a system for learning *Horn* clauses from data expressed as relations. To avoid ambiguity, here we assume H as a set of Horn clauses which contain only atoms in their bodies.

Thus, the relation (3) is rewritten as

$$E \subseteq M_{B \cup H}$$

where $M_{B \cup H}$ is the least model of $B \cup H$. Note that $B \cup H$ is a definite logic program and is always consistent. By this fact, inductive equivalence in FOIL is defined as follows.

Definition 4.1 (inductive equivalence in FOIL) Two Datalog programs B_1 and B_2 are *inductively equivalent in* FOIL if it holds that $O \subseteq M_{B_1 \cup H}$ iff $O \subseteq M_{B_2 \cup H}$ for any observation O and for any hypothesis H.

Let H be a set of ground definite clauses and M a set of ground atoms. Then, define

$$T_H(M) = \{A \mid A \leftarrow A_1, \ldots, A_n \text{ is in } H \text{ and } \{A_1, \ldots, A_n\} \subseteq M\}.$$

Theorem 4.1 *(condition for inductive equivalence in* FOIL*) Let B_1 and B_2 be two Datalog programs. Then, B_1 and B_2 are inductively equivalent in* FOIL *iff $B_1 \equiv_w B_2$.*

Proof For any O and H, $O \subseteq M_{B_1 \cup H}$ iff $O \subseteq M_{B_2 \cup H}$
$\Leftrightarrow M_{B_1 \cup H} = M_{B_2 \cup H}$. $(*)$
Putting $H = \emptyset$, $(*)$ implies $M_{B_1} = M_{B_2}$. Hence, $B_1 \equiv_w B_2$.

Conversely, if $B_1 \equiv_w B_2$, then $M_{B_1} = M_{B_2}$. Suppose any set H of ground clauses such that $H = \{A \leftarrow A_1, \ldots, A_n \mid A \in E\}$. Then, $M_{B_1} \cup T_H(M_{B_1}) = M_{B_2} \cup T_H(M_{B_2})$. Since $M_{B_i} \cup T_H(M_{B_i}) = M_{B_i \cup H}$ for $i = 1, 2$, $M_{B_1 \cup H} = M_{B_2 \cup H}$. Hence, $O \subseteq M_{B_1 \cup H}$ iff $O \subseteq M_{B_2 \cup H}$ for any O and H. □

Example 4.1 Two programs

$$B_1 = \{p(x) \leftarrow q(x), \quad r(a) \leftarrow\},$$
$$B_2 = \{r(a) \leftarrow\}$$

have the same least model $\{r(a)\}$, thereby weakly equivalent. Hence, B_1 and B_2 are inductively equivalent in FOIL.

In Example 4.1, B_1 and B_2 are not inductively equivalent in explanatory induction in general. In fact, for the observation $O = \{p(a)\}$, the hypothesis $H = \{q(x) \leftarrow r(x)\}$ explains $p(a)$ in B_1, but not in B_2. The hypothesis H is not produced in FOIL, however, because the predicate q appearing in the head does not appear in O.

4.2 GOLEM

GOLEM (Muggleton and Feng 1990) realizes explanatory induction in definite logic programs. It uses the algorithm of relative least generalization under subsumption (Plotkin 1971). We first review basic terms and results. A clause C_1 *subsumes* another clause C_2 *relative to* a program B, denoted by $C_1 \succeq_B C_2$, if there is a substitution θ such that $B \models C_1\theta \rightarrow C_2$. A clause D is a *relative least generalization under subsumption* (*rlgs*) of C_1 and C_2 with respect to B if D is the least upper bound of C_1 and C_2 under the ordering $\succeq_B$ over the clausal language. The *rlgs* does not always exist but exists when B is a set of ground atoms (Nienhuys-Cheng and Wolf 1997).

Given a definite logic program B and a set O of ground facts, GOLEM constructs a hypothesis H as follows:

$$\begin{aligned} & B \cup H \models O \\ \Leftrightarrow \quad & H \models B \rightarrow O \\ \Leftrightarrow \quad & \models H \rightarrow (\neg B \vee O). \end{aligned}$$

At this point, GOLEM replaces B with the conjunction of ground atoms included in a finite subset of the least model M_B of B. For simplicity reasons, we suppose that the least model M_B is finite and replace B with M_B. Let $O = \{ A_1, \ldots, A_k \}$. Then,

$$H \rightarrow \neg M_B \vee O$$

where

$$\neg M_B \vee O = (A_1 \vee \neg M_B) \wedge \cdots \wedge (A_k \vee \neg M_B)$$

with $\neg M_B = \bigvee_{A_i \in M_B} \neg A_i$. Next, the $rlgs$ of O with respect to M_B (written as $rlgs(M_B, O)$) is computed as the *least generalization under subsumption* (lgs) of clauses $(A_1 \vee \neg M_B), \ldots, (A_k \vee \neg M_B)$ (written as $lgs(A_1 \vee \neg M_B, \ldots, A_k \vee \neg M_B)$). A hypothesis H is then put as

$$H = rlgs(M_B, O)$$

which is a set of definite clauses.

Inductive equivalence in GOLEM is now defined as follows.

Definition 4.2 (inductive equivalence in GOLEM) Let B_1 and B_2 be two definite logic programs such that each program has the least model as a finite set. Then, B_1 and B_2 are *inductively equivalent in* GOLEM if $rlgs(M_{B_1}, O) = rlgs(M_{B_2}, O)$ for any set O of ground facts.

We then have the following result.

Theorem 4.2 *(condition for inductive equivalence in* GOLEM*) Let B_1 and B_2 be two definite logic programs. Then, B_1 and B_2 are inductively equivalent in* GOLEM *iff* $B_1 \equiv_w B_2$.

Proof Suppose that B_1 and B_2 are inductively equivalent in GOLEM. Then, for any set $O = \{ A_1, \ldots, A_k \}$ of ground facts, $rlgs(M_{B_1}, O) = rlgs(M_{B_2}, O)$ implies $lgs(A_1 \vee \neg M_{B_1}, \ldots, A_k \vee \neg M_{B_1}) = lgs(A_1 \vee \neg M_{B_2}, \ldots, A_k \vee \neg M_{B_2})$. Put $O = \{A\}$ for any ground atom A. Then, $lgs(A \vee \neg M_{B_1}) = lgs(A \vee \neg M_{B_2})$ implies $A \vee \neg M_{B_1} = A \vee \neg M_{B_2}$ thereby $M_{B_1} = M_{B_2}$. Hence, $B_1 \equiv_w B_2$.

Conversely, if $B_1 \equiv_w B_2$, $M_{B_1} = M_{B_2}$. Then, for any set $O = \{ A_1, \ldots, A_k \}$ of ground facts, $lgs(A_1 \vee \neg M_{B_1}, \ldots, A_k \vee \neg M_{B_1}) = lgs(A_1 \vee \neg M_{B_2}, \ldots, A_k \vee \neg M_{B_2})$, so $rlgs(M_{B_1}, O) = rlgs(M_{B_2}, O)$. Hence, the result holds. □

Example 4.2 Consider two programs:

$$\begin{aligned} B_1 = \{ \quad & has\_wings(joe) \leftarrow bird(joe), \\ & bird(tweety) \leftarrow, \\ & bird(polly) \leftarrow \}. \\ B_2 = \{ \quad & bird(tweety) \leftarrow, \\ & bird(polly) \leftarrow \}. \end{aligned}$$

Given the observation $O = \{ flies(tweety), flies(polly) \}$, both $rlgs(M_{B_1}, O)$ and $rlgs(M_{B_2}, O)$ contain the single clause:

$$flies(x) \leftarrow bird(x).$$

This means that the first clause of B_1 is of no use for induction in GOLEM. Note that B_1 and B_2 are weakly equivalent, but they are not logically equivalent.

In the process of constructing inductive hypothesis H, GOLEM approximates B to a finite subset of M_B. However, $rlgs(B, O) \neq rlgs(M_B, O)$ in general. In fact, in Example 4.2, given $O = \{ has\_wing(joe) \}$, the hypothesis $H = \{ bird(joe) \}$ is obtained in B_1 but not in B_2. This means that some hypotheses which are computed under $rlgs$ might be lost by GOLEM.

4.3 PROGOL

PROGOL is also known as a Horn ILP system which realizes explanatory induction. It is based on the inverse entailment algorithm developed in (Muggleton 1995). Given a Horn logic program B and a ground Horn clause O as an observation, suppose a Horn clause H satisfying

$$B \cup \{H\} \models O.$$

By inverting the entailment relation it becomes

$$B \cup \{\neg O\} \models \neg H.$$

Put $\neg bot(B, O)$ as the conjunction of ground literals which are true in every model of $B \cup \{\neg O\}$. Then, a clause H is induced by *inverse entailment* (IE) if $H \models bot(B, E)$ where $bot(B, E)$ is a clause called a *bottom clause*.[4]

Inductive equivalence in PROGOL is defined as follows.

Definition 4.3 (inductive equivalence in PROGOL) Two Horn logic programs B_1 and B_2 are *inductively equivalent in* PROGOL if $bot(B_1, O) = bot(B_2, O)$ for any ground Horn clause O.

Then, we have the following result.

Theorem 4.3 *(condition for inductive equivalence in* PROGOL*) Two Horn logic programs B_1 and B_2 are inductively equivalent under* PROGOL *iff $B_1 \equiv B_2$.*

Proof B_1 and B_2 are inductively equivalent under PROGOL iff $bot(B_1, O) = bot(B_2, O)$ for any O. Then, $\neg\, bot(B_1, O) = \neg\, bot(B_2, O)$, and $B_1 \cup \{\neg O\} \models L$ iff $B_2 \cup \{\neg O\} \models L$ for any ground Horn clause O and for any ground literal L. Put $O = \leftarrow A_1, \ldots, A_n$. Then, $B_1 \cup \{A_1, \ldots, A_n\} \models L$ iff $B_2 \cup \{A_1, \ldots, A_n\} \models L$ for any $\{ A_1, \ldots, A_n \}$. Thus, for any finite set F of ground atoms, $B_1 \cup F \models L$ iff $B_2 \cup F \models L$. So, $B_1 \models F \supset L$ iff $B_2 \models F \supset L$ for any finite set F of ground atoms and any ground literal L. This implies $B_1 \equiv B_2$. Conversely, $B_1 \equiv B_2$ implies $bot(B_1, O) = bot(B_2, O)$, hence the result holds. □

[4] Strictly speaking, PROGOL does not produce every clause satisfying the relation $H \models bot(B, E)$ and is in this sense *incomplete* (Badea and Stanciu 1999). But here we proceed our discussion by assuming an ideal algorithm which computes every H satisfying the relation. *CF-induction* (Inoue 2004) realizes a sound and complete induction algorithm based on IE in full clausal theories.

In PROGOL, weak equivalence of two programs is not sufficient for inductive equivalence.

Example 4.3 Consider two programs:

$$B_1 = \{\ white\_swan(c) \leftarrow \}.$$
$$B_2 = \{\ abnormal(x) \leftarrow black\_swan(x),$$
$$\quad white\_swan(c) \leftarrow \}.$$

Given the observation $O = \leftarrow black\_swan(c)$, it becomes

$$\neg\, bot(B_1, O) = white\_swan(c) \wedge black\_swan(c).$$

Then,

$$H_1 = \leftarrow white\_swan(x), black\_swan(x)$$

becomes a hypothesis satisfying $H_1 \models bot(B_1, E)$. By contrast,

$$\neg\, bot(B_2, O) = white\_swan(c) \wedge black\_swan(c) \wedge abnormal(c).$$

Then,

$$H_2 = \leftarrow abnormal(x)$$

becomes a hypothesis satisfying $H_2 \models bot(B_2, E)$. Note that $B_1 \not\equiv B_2$ but $B_1 \equiv_w B_2$.

4.4 CLAUDIEN

The system CLAUDIEN (De Raedt and Bruynooghe 1993; De Raedt and Dehaspe 1997b) realizes descriptive induction under the *completion* semantics (Clark 1978). Given a definite logic program B and a set O of definite clauses, CLAUDIEN produces a set H of clauses satisfying

$$Comp(B \cup O) \models H$$

where $Comp$ represents Clark's predicate completion.

Example 4.4 Let $B = \{\, human(s) \,\}$ and $O = \{\, mortal(s) \,\}$. Then, the following clauses are all possible solutions:

$$H_1 = \{\ mortal(x) \leftarrow human(x)\ \},$$
$$H_2 = \{\ human(x) \leftarrow mortal(x)\ \},$$
$$H_3 = \{\ human(x) \vee mortal(x) \leftarrow \}.$$

Note that in descriptive induction it is assumed that the universe defined by an observation together with a background theory is completely specified (De Raedt and Lavrač 1993).

Definition 4.4 (inductive equivalence in CLAUDIEN) Let B_1 and B_2 be two definite logic programs. Then, B_1 and B_2 are *inductively equivalent in* CLAUDIEN if it holds that

$$Comp(B_1 \cup O) \models H \quad \text{iff} \quad Comp(B_2 \cup O) \models H$$

for any observation O and for any hypothesis H such that $B_1 \cup O$ and $B_2 \cup O$ are consistent.

Table 1 Comparison of Horn ILP Systems

System	language	induction	condition
FOIL	Datalog	ExplInd	$B_1 \equiv_w B_2$
GOLEM	definite LP	ExplInd	$B_1 \equiv_w B_2$
PROGOL	Horn LP	ExplInd	$B_1 \equiv B_2$
CLAUDIEN	definite LP	DesInd	$B_1 \equiv B_2$

Theorem 4.4 *(condition for inductive equivalence in* CLAUDIEN*) Two definite logic programs B_1 and B_2 are inductively equivalent in* CLAUDIEN *iff $B_1 \equiv B_2$.*

Proof It is shown that $B_1 \equiv B_2$ iff $Comp(B_1 \cup O) \equiv Comp(B_2 \cup O)$ for any clausal theory O. The proof is similar to Proposition 2.2. □

The results of Section 4 are summarized in Table 1. Observe that the conditions of inductive equivalence in FOIL and GOLEM are weaker than the condition of inductive equivalence in explanatory induction (Theorem 3.2). This is due to the fact that these systems impose some restrictions on the syntax of background theories, observations and hypotheses. By contrast, the condition of inductive equivalence in PROGOL and CLAUDIEN is identical to the one in clausal logic.

5 Inductive Equivalence in Nonmonotonic Logic Programs

5.1 Nonmonotonic Logic Programs

Nonmonotonic logic programs are logic programs with *negation as failure* (Baral and Gelfond 1994). We consider the class of extended disjunctive programs (Gelfond and Lifschitz 1991) in this paper. An *extended disjunctive program* (EDP) (or simply a *program*) is a set of *rules* of the form:

$$L_1 \,;\, \cdots \,;\, L_l \;\leftarrow\; L_{l+1}, \ldots, L_m, \mathit{not}\, L_{m+1}, \ldots, \mathit{not}\, L_n \quad (n \geq m \geq l \geq 0) \qquad (4)$$

where each L_i is a positive/negative literal, i.e., A or $\neg A$ for an atom A, and *not* is *negation as failure* (NAF). $\mathit{not}\, L$ is called an *NAF-literal*. The symbol ";" represents disjunction. The left-hand side of "$\leftarrow$" is the *head*, and the right-hand side is the *body*. For each rule r of the form (4), $head(r)$, $body^+(r)$ and $body^-(r)$ denote the sets of literals $\{L_1, \ldots, L_l\}$, $\{L_{l+1}, \ldots, L_m\}$, and $\{L_{m+1}, \ldots, L_n\}$, respectively. Also, $not\_body^-(r)$ denotes the set of NAF-literals $\{\mathit{not}\, L_{m+1}, \ldots, \mathit{not}\, L_n\}$. A disjunction of literals and a conjunction of (NAF-)literals in a rule are identified with its corresponding sets of literals. A rule r is *disjunctive* if $head(r)$ contains more than one literal. A rule r is a *constraint* if $head(r) = \emptyset$; and r is a *fact* if $body(r) = \emptyset$. A program is *basic* if no rule contains NAF-literals. A program, rule, or literal is *ground* if it contains no variable. A *propositional* program is a finite set of ground rules. A program P with variables is a shorthand of its *ground instantiation* $ground(P)$, the (possibly infinite) set of ground rules obtained from P by substituting variables in P by elements of its Herbrand universe in every possible way. A program is called an *extended logic program* (ELP) if it contains no disjunctive rule. An ELP is called a *normal logic program* (NLP) if every literal L_i appearing in the program is an atom.

A primary difference between nonmonotonic logic programs and clausal theories is that a rule (4) is *not* a clause even if it contains no NAF-literal. For instance, a rule

$L_1 \leftarrow L_2$ has meaning different from $\neg L_2 \leftarrow \neg L_1$ or $L_1 \vee \neg L_2$. The rule (4) is interpreted as an *inference rule* rather than an implication formula (Gelfond and Lifschitz 1991). Thus, induction in nonmonotonic logic programs is different from induction in clausal theories.

The semantics of an EDP is defined by the *answer set semantics* (Gelfond and Lifschitz 1991). The *literal base* Lit is the set of all ground literals in the language of a program. Suppose a program P and a set of literals $S(\subseteq Lit)$. Then, the *reduct* P^S is the program which contains the ground rule $head(r) \leftarrow body^+(r)$ iff there is a rule r in $ground(P)$ such that $body^-(r) \cap S = \emptyset$. Given a basic program P, let S be a set of ground literals that is (i) *closed* under P, i.e., for every ground rule r in $ground(P)$, $body(r) \subseteq S$ implies $head(r) \cap S \neq \emptyset$; and (ii) *logically closed*, i.e., it is either consistent or equal to Lit. An *answer set* of a basic program P is a minimal set S satisfying both (i) and (ii). Given an EDP P and a set S of ground literals, S is an *answer set* of P if S is an answer set of P^S. A program has none, one, or multiple answer sets in general. The set of all answer sets of P is written as $AS(P)$. Here $AS(P)$ is an *antichain* set, i.e., no element $S \in AS(P)$ is a proper subset of another element $T \in AS(P)$. An answer set is *consistent* if it is not Lit. A program P is *consistent* if it has a consistent answer set; otherwise, P is *inconsistent*. In normal logic programs, answer sets are also called *stable models* (Gelfond and Lifschitz 1988). A set S of ground literals *satisfies* a ground rule r if either $S \cap head(r) \neq \emptyset$, $body^+(r) \setminus S \neq \emptyset$ or $body^-(r) \cap S \neq \emptyset$. When a rule r contains variables, S satisfies r if S satisfies every ground instance of r. S satisfies a set R of rules if S satisfies every rule in R. A program P *satisfies* a set R of rules if every answer set of P satisfies every rule in R. A program P is *consistent* if it has an answer set; otherwise P is *inconsistent*.

Example 5.1 Let P be the program:

$$
\begin{aligned}
&p(x) \leftarrow not\ q(x),\\
&q(x) \leftarrow not\ p(x),\\
&r(a) \leftarrow
\end{aligned}
$$

where $AS(P) = \{\{p(a), r(a)\}, \{q(a), r(a)\}\}$. Then, every answer set satisfies the rule $p(a)\,;\, q(a) \leftarrow$, while $p(a) \leftarrow r(a)$ is satisfied by $\{p(a), r(a)\}$, but not by $\{q(a), r(a)\}$.

5.2 Inductive Equivalence between EDPs

Four different induction frameworks in Section 2.2 are applied to induction in nonmonotonic logic programs. In this section, we consider the following problem setting:

- a background theory B is given as an EDP under the answer set semantics $SEM(B) = AS(B)$,
- an observation O is a set of rules,
- a hypothesis H is a set of rules.

Note that in case of nonmonotonic logic programs, a background theory B could be inconsistent. In this case, the introduction of H to B makes $B \cup H$ consistent. This is the difference from the case of monotonic background theories. In case of monotonic theories, an inconsistent B cannot become consistent by introducing any H.

Example 5.2 Let B be the program:

$$p \leftarrow not\ p.$$

Then, B is inconsistent, i.e., $AS(B) = \emptyset$. Putting $H = \{p \leftarrow\}$, $B \cup H$ becomes consistent, i.e., $AS(B \cup H) = \{\{p\}\}$.

Four different definitions of inductive equivalence are then considered under the answer set semantics.

Definition 5.1 (inductive equivalence in different frameworks of induction) Let B_1 and B_2 be two programs having the same literal base Lit. Then,

1. B_1 and B_2 are *inductively equivalent* under the answer set semantics in cautious induction (written $B_1 \equiv^{AS}_{\mathsf{CauInd}} B_2$) if for any O and any H, O is satisfied by every $S \in AS(B_1 \cup H)$ iff O is satisfied by every $S \in AS(B_2 \cup H)$, where $B_1 \cup H$ and $B_2 \cup H$ are consistent.
2. B_1 and B_2 are *inductively equivalent* under the answer set semantics in brave induction (written $B_1 \equiv^{AS}_{\mathsf{BraInd}} B_2$) if for any O and any H, O is satisfied by some $S \in AS(B_1 \cup H)$ iff O is satisfied by some $S \in AS(B_2 \cup H)$, where $B_1 \cup H$ and $B_2 \cup H$ are consistent.
3. B_1 and B_2 are *inductively equivalent* under the answer set semantics in learning from satisfiability (written $B_1 \equiv^{AS}_{\mathsf{LFS}} B_2$) if for any O and any H, $B_1 \cup H \cup O$ is consistent iff $B_2 \cup H \cup O$ is consistent.
4. B_1 and B_2 are *inductively equivalent* under the answer set semantics in descriptive induction (written $B_1 \equiv^{AS}_{\mathsf{DesInd}} B_2$) if for any O and any H, H is satisfied by every $S \in AS(B_1 \cup O)$ iff H is satisfied by every $S \in AS(B_2 \cup O)$, where $B_1 \cup O$ and $B_2 \cup O$ are consistent.

In each case, we say that a hypothesis H *covers* (or *explains*) O with respect to B under the answer set semantics in the induction framework I. Here, I is one of the four induction frameworks presented above.

Next we provide a program transformation which is useful for subsequent discussion. Given a set O of ground rules, any rule r in O is transformed to the set Ω of rules:

$$\begin{array}{ll} G_r \leftarrow L_i & \text{for every } L_i \in head(r), \\ G_r \leftarrow not\ L_j & \text{for every } L_j \in body^+(r), \\ G_r \leftarrow L_k & \text{for every } L_k \in body^-(r), \end{array}$$

where G_r is a new ground atom appearing nowhere in B and uniquely associated with each r. With this setting, the next result holds.

Proposition 5.1 *O is satisfied by an answer set of $B \cup H$ iff for any $r \in O$, G_r is included in an answer set of $B \cup H \cup \Omega$.*

Proof O is satisfied by an answer set S of $B \cup H$
iff for any r in O, either $S \cap head(r) \neq \emptyset$, $body^+(r) \setminus S \neq \emptyset$ or $body^-(r) \cap S \neq \emptyset$ for some $S \in AS(B \cup H)$
iff $B \cup H \cup \Omega$ has an answer set $T = S \cup \{G_r \mid r \in O\}$ for some $S \in AS(B \cup H)$. □

Thus, any observation O as a set of rules is instantiated to its ground instances $ground(O)$, which is then transformed to a semantically equivalent observation as a set of ground atoms.

Proposition 5.2 *(relations between different inductive equivalences) The following relations hold between equivalence relations in different induction.*

1. $B_1 \equiv^{AS}_{\mathsf{CautInd}} B_2$ *implies* $B_1 \equiv^{AS}_{\mathsf{BravInd}} B_2$.
2. $B_1 \equiv^{AS}_{\mathsf{BravInd}} B_2$ *iff* $B_1 \equiv^{AS}_{\mathsf{LFS}} B_2$.
3. $B_1 \equiv^{AS}_{\mathsf{CautInd}} B_2$ *iff* $B_1 \equiv^{AS}_{\mathsf{DesInd}} B_2$.

Proof The results (1) and (3) follow from definitions. We show (2). If $B_1 \equiv^{AS}_{\mathsf{LFS}} B_2$, for any H and any O, $B_1 \cup H \cup \Omega \cup \{ \leftarrow not\, G_r \mid r \in ground(O) \}$ is consistent iff $B_2 \cup H \cup \Omega \cup \{ \leftarrow not\, G_r \mid r \in ground(O) \}$ is consistent. Put $U = \{ G_r \mid r \in ground(O) \}$. Then, for any H and any U, $U \subseteq S$ for some consistent answer set S of $B_1 \cup H \cup \Omega$ iff $U \subseteq T$ for some consistent answer set T of $B_2 \cup H \cup \Omega$. Hence, $B_1 \equiv^{AS}_{\mathsf{BravInd}} B_2$. The only-if part clearly holds. □

Proposition 5.2(2) presents that the notions of inductive equivalence in brave induction and learning from satisfiability coincide under the answer set semantics.

We proceed to build conditions for inductive equivalence between EDPs.

Theorem 5.3 *(condition for inductive equivalence under AS in cautious induction) Let B_1 and B_2 be any EDPs. Then, $B_1 \equiv_s B_2$ implies $B_1 \equiv^{AS}_{\mathsf{CautInd}} B_2$. The converse implication also holds for any H such that $AS(B_1 \cup H)$ and $AS(B_2 \cup H)$ are finite sets.*[5]

Proof If $B_1 \equiv_s B_2$, $AS(B_1 \cup H) = AS(B_2 \cup H)$ holds for any set H of rules. In this case, O is satisfied by every answer set of $B_1 \cup H$ iff O is satisfied by every answer set of $B_2 \cup H$ for any O and H such that $B_1 \cup H$ and $B_2 \cup H$ are consistent. Hence, B_1 and B_2 are inductively equivalent in cautious induction.

Conversely, suppose that B_1 and B_2 are inductively equivalent in cautious induction. Then, it holds that O is satisfied in every answer set of $B_1 \cup H$ iff O is satisfied in every answer set of $B_2 \cup H$ for any O and any H such that $B_1 \cup H$ and $B_2 \cup H$ are consistent. Suppose that there is a set S such that $S \in AS(B_1 \cup H) \setminus AS(B_2 \cup H)$ for some H. For any answer set T_i of $B_2 \cup H$, put

$$\bigcup_i (S \setminus T_i) = U \quad \text{and} \quad \bigcup_i (T_i \setminus S) = V.$$

For some non-empty finite subset of $U' \subseteq U$ and $V' \subseteq V$, construct the constraint

$$C : \leftarrow U',\ not\, V'$$

where U' or V' is identified with the conjunction of literals included in each set. By $U' \subseteq S$ and $V' \cap S = \emptyset$, S does not satisfy C.

If every answer set T_i of $B_2 \cup H$ satisfies C, this contradicts the assumption that B_1 and B_2 are inductively equivalent. Else if some answer set T_i of $B_2 \cup H$ does not satisfy C, $U' \subseteq T_i$ and $V' \cap T_i = \emptyset$. For every such T_i, either $S \setminus T_i \neq \emptyset$ or $T_i \setminus S \neq \emptyset$

[5] At the moment, the result is open when a program has an infinite number of answer sets.

holds by $S \notin AS(B_2 \cup H)$. For every T_i satisfying $S \setminus T_i \neq \emptyset$, take one literal L_i from $S \setminus T_i$ and collect such a literal from each T_i. Put the collection as W_1:

$$W_1 = \bigcup_i \{ L_i \mid L_i \in S \setminus T_i \text{ where } S \setminus T_i \neq \emptyset \}.$$

Similarly, for every T_i satisfying $T_i \setminus S \neq \emptyset$, take one literal L_i from $T_i \setminus S$ and collect such a literal from each T_i. Put the collection as W_2:

$$W_2 = \bigcup_i \{ L_i \mid L_i \in T_i \setminus S \text{ where } T_i \setminus S \neq \emptyset \}.$$

Here, W_1 and W_2 are finite set, because $B_2 \cup H$ has a finite number of answer sets. Suppose the constraint

$$D: \leftarrow U, W_1, \mathit{not}\, V, \mathit{not}\, W_2 .$$

As $W_1 \not\subseteq T_i$ or $W_2 \cap T_i \neq \emptyset$ holds for any T_i, T_i satisfies D. Thus, D is satisfied by every answer set of $B_2 \cup H$. On the other hand, $W_1 \subseteq S$ and $W_2 \cap S = \emptyset$ imply that S does not satisfy D. Then, D is not satisfied by some answer set of $B_1 \cup H$. This contradicts the assumption that B_1 and B_2 are inductively equivalent. □

Theorem 5.4 *(condition for inductive equivalence under AS in brave induction) Let B_1 and B_2 be any EDPs. Then, $B_1 \equiv^{AS}_{\mathsf{Bral nd}} B_2$ iff $B_1 \equiv_s B_2$.*

Proof When B_1 and B_2 are inductively equivalent in brave induction, it holds that O is satisfied in an answer set of $B_1 \cup H$ iff O is satisfied in an answer set of $B_2 \cup H$ for any O and any H such that $B_1 \cup H$ and $B_2 \cup H$ are consistent. Then, for any set O of ground literals, $O \subseteq S$ for an answer set S of $B_1 \cup H$ iff $O \subseteq T$ for an answer set T of $B_2 \cup H$. Putting $O = S$, S is an answer set of $B_1 \cup H$ iff $S \subseteq T$ for an answer set T of $B_2 \cup H$ $(*)$. Putting $O = T$, T is an answer set of $B_2 \cup H$ iff $T \subseteq S'$ for an answer set S' of $B_1 \cup H$ $(\dagger)$. By $(*)$ and $(\dagger)$, S is an answer set of $B_1 \cup H$ iff $S \subseteq T \subseteq S'$ for an answer set T of $B_2 \cup H$ and for an answer set S' of $B_1 \cup H$. Since $AS(B_1 \cup H)$ is an antichain set, $S = T$. Thus, S is an answer set of $B_1 \cup H$ iff S is an answer set of $B_2 \cup H$ for any H. Hence, B_1 and B_2 are strongly equivalent.

Conversely, if $B_1 \equiv_s B_2$, $AS(B_1 \cup H) = AS(B_2 \cup H)$ for any set H of rules. Then, O is satisfied in an answer set of $B_1 \cup H$ iff O is satisfied in an answer set of $B_2 \cup H$ for any set O of rules. Hence, $B_1 \equiv^{AS}_{\mathsf{Bral nd}} B_2$ holds. □

Theorem 5.5 *(condition for inductive equivalence under AS in learning from satisfiability) Let B_1 and B_2 be any EDPs. Then, $B_1 \equiv^{AS}_{\mathsf{LFS}} B_2$ iff $B_1 \equiv_s B_2$.*

Proof The result holds by Theorems 5.2(2) and 5.4. □

The complexity of testing strong equivalence of two propositional EDPs is coNP-complete (Turner 2003). Hence we have the next result.

Proposition 5.6 *(complexity for deciding inductive equivalence between EDPs) Deciding inductive equivalence of two propositional EDPs is coNP-complete under the answer set semantics in four different induction frameworks.*

5.3 Inductive Equivalence in Nonmonotonic ILP Systems

In this section, we investigate inductive equivalence in two nonmonotonic ILP systems.

5.3.1 Induction of Stable Models

Otero (2001) characterizes induction problems in normal logic programs (NLPs) under the stable model semantics. Recall that an NLP is a set of rules of the form:

$$A_0 \leftarrow A_1, \ldots, A_m, \mathit{not}\, A_{m+1}, \ldots, \mathit{not}\, A_n \quad (n \geq m \geq 0) \tag{5}$$

where each A_i is an atom. Answer sets coincide with stable models in NLPs, so that Otero's framework is considered a special case of induction under the answer set semantics. Otero introduces different types of induction for positive/negative observations, but here we consider the so-called *induction from non-complete sets* which is the usual ILP setting for positive observations.

Suppose a background theory B as an NLP, and a set O of ground atoms as a positive observation such that O is not satisfied by B. The goal is to find a set H of rules satisfying the condition that O is satisfied by every stable model of $B \cup H$. Thus, Otero's framework realizes cautious induction in nonmonotonic logic programs. An interpretation M is a *monotonic model* of an NLP if M satisfies every rule in B. A stable model is a monotonic model, but not vice versa. Given an observation O, an interpretation M is an *extension* of O iff $O \subseteq M$. He then captures the computation of H as an extension M of O that becomes a stable model of $B \cup M$. That is, $H = M$ satisfying $O \subseteq M$ and $M \in AS(B \cup M)$ becomes a solution. Note that in this definition a hypothesis H is given as a set of ground atoms.

Let $ISM(B, O)$ be the collection of H defined as above. Then, inductive equivalence in *induction of stable models* (ISM) is defined as follows.

Definition 5.2 (inductive equivalence in ISM) Two NLPs B_1 and B_2 are *inductively equivalent in ISM* if $ISM(B_1, O) = ISM(B_2, O)$ for any set O of ground atoms.

Proposition 5.7 *(Otero 2001) Given an NLP B, M is a monotonic model of B iff M is a stable model of $B \cup M$.*

Let $MonMod(B)$ be the set of monotonic models of B. Then we have the following result.

Theorem 5.8 *(condition for inductive equivalence in ISM) Two NLPs B_1 and B_2 are inductively equivalent in ISM iff $MonMod(B_1) = MonMod(B_2)$.*

Proof Suppose that B_1 and B_2 are inductively equivalent in ISM. For any $M \in ISM(B_1, O)$, M is a stable model of $B_1 \cup M$ and a monotonic model of B_1 (Proposition 5.7). Then, $ISM(B_1, O) = ISM(B_2, O)$ implies $MonMod(B_1) = MonMod(B_2)$. Conversely, if $MonMod(B_1) = MonMod(B_2)$, for any set M of atoms, M is a stable model of $B_1 \cup M$ iff M is a stable model of $B_2 \cup M$. Then, for any set O of ground atoms, $M (\supseteq O)$ is a stable model of $B_1 \cup M$ iff M is a stable model of $B_2 \cup M$. Hence, $ISM(B_1, O) = ISM(B_2, O)$. □

Example 5.3 Let $B_1 = \{ p \leftarrow \mathit{not}\, q \}$ and $B_2 = \{ q \leftarrow \mathit{not}\, p \}$. For $O = \{p\}$, put its extension as $M = \{p\}$. Then, $H = \{ p \leftarrow \}$ becomes a solution in both B_1 and B_2. Note that $B_1 \not\equiv_w B_2$ but $MonMod(B_1) = MonMod(B_2)$.

Since $AS(B) \subseteq MonMod(B)$, the above result implies that inductive equivalence in ISM does not require the condition of strong nor weak equivalence. It is worth noting

that induction in ISM can be reformulated using classical logic. Given an NLP B, consider a clausal theory $Cl(B)$ which is obtained from B by replacing every NAF-literal $not\ A$ in (5) with a negative literal $\neg A$. Then, monotonic models of B coincide with Herbrand models of $Cl(B)$. Thus, the inductive equivalence in ISM is translated into the problem of inductive equivalence under the classical semantics, and Theorem 5.8 implies that $ISM(B_1, E) = ISM(B_2, E)$ iff $Cl(B_1) \equiv Cl(B_2)$.

5.3.2 Brave Induction from Answer Sets

Sakama and Inoue (2009b) introduce an algorithm for brave induction in nonmonotonic logic programs. Given a background theory B as an EDP and a set O of ground literals as an observation, the algorithm BRAIN^{not} computes a set H of rules as a hypothesis. Before presenting an algorithm, a couple of notions are in order. Given a literal L, $pred(L)$ and $const(L)$ represent the predicate of L and the constants appearing in L, respectively. Let L_0 be a ground literal and S a set of ground literals. Then, $L_1 \in S$ is *relevant* to L_0 if either (i) $const(L_0) \cap const(L_1) \neq \emptyset$, or (ii) for some literal $L_2 \in S$, $const(L_1) \cap const(L_2) \neq \emptyset$ and L_2 is relevant to L_0. Otherwise, $L_1 \in S$ is *irrelevant* to L_0. Rules $r_1, \ldots, r_k$ are *comparable* if there is a predicate appearing in every $head(r_1), \ldots, head(r_k)$.

BRAIN^{not} constructs hypotheses in the following two steps.[6] First, for a consistent answer set S of B, construct a finite and consistent set R_S of ground rules satisfying the following conditions. For any rule $r \in R_S$,

1. $head(r) \subseteq O$ and for any $L \in O$, there is a rule $r \in R_S$ such that $head(r) = \{L\}$,
2. $body^+(r) \subseteq P$ where $P = \{\ L \mid L \in S$ and L is relevant to the literal in $head(r)\ \}$,
3. $body^-(r) \subseteq N$ where $N = \{\ L \mid L \in Lit \setminus (S \cup \Theta)$ and L is relevant to the literal in $head(r)$ and appears in $ground(B)\ \}$

where $\Theta = \{\ L \mid L \in Lit$ and $pred(L)$ appears in $O\ \}$. In the second and third conditions, we put $body^+(r) = P$ and $body^-(r) = N$ if P and N are finite sets.

Second, for the set R_S of rules obtained as above, R_S is partitioned as $R_S = R_1 \cup \cdots \cup R_n$ where each R_i $(1 \leq i \leq n)$ is a comparable set of ground rules. Then, the least generalization under subsumption of each R_i is computed and collected as[7]

$$lgs(R_S) = \{\ lgs(R_1),\ \ldots,\ lgs(R_n)\}.$$

$lgs(R_S)$ is a *solution* of brave induction if $B \cup lgs(R_S)$ is consistent.

Example 5.4 Suppose the background theory B:

$$innocent(x) \leftarrow not\ guilty(x),$$
$$guilty(x) \leftarrow not\ innocent(x),$$
$$suspect(a) \leftarrow,$$
$$suspect(b) \leftarrow,$$

[6] The algorithm in (Sakama and Inoue 2009b) has additional two steps for constructing weak hypotheses and optimization, but we omit these steps here for simplicity reasons.

[7] The lgs of rules is computed in the same way as the case of clauses.

which has four answer sets:

$$S_1 = \{suspect(a), suspect(b), guilty(a), guilty(b)\},$$
$$S_2 = \{suspect(a), suspect(b), guilty(a), innocent(b)\},$$
$$S_3 = \{suspect(a), suspect(b), innocent(a), guilty(b)\},$$
$$S_4 = \{suspect(a), suspect(b), innocent(a), innocent(b)\}.$$

Given the observation $O = \{charged(a), charged(b)\}$, the set of ground rules

$$R_{S_1} = \{\ charged(a) \leftarrow suspect(a),\ guilty(a),\ not\ innocent(a),$$
$$charged(b) \leftarrow suspect(b),\ guilty(b),\ not\ innocent(b)\ \}$$

is constructed using the answer set S_1. The lgs of R_{S_1} becomes

$$lgs(R_{S_1}) = \{\ charged(x) \leftarrow suspect(x),\ guilty(x),\ not\, innocent(x)\ \},$$

then $B \cup lgs(R_{S_1})$ has the answer set $S_1 \cup \{charged(a), charged(b)\}$ which satisfies O.

By the definition, different hypotheses are constructed by different answer sets. Now we define inductive equivalence in BRAIN^{not} as follows.

Definition 5.3 (inductive equivalence in BRAIN^{not}) Two EDPs B_1 and B_2 are *inductively equivalent in* BRAIN^{not} if for any set O of ground literals, $lgs(R_S) = lgs(R_T)$ holds for some consistent $S \in AS(B_1)$ and some consistent $T \in AS(B_2)$ such that $B_1 \cup lgs(R_S)$ and $B_2 \cup lgs(R_T)$ are consistent.

Then we have the following result.

Theorem 5.9 *(condition for inductive equivalence in* BRAIN^{not}*) Two EDPs B_1 and B_2 are inductively equivalent in* BRAIN^{not} *iff $B_1 \equiv_s B_2$.*

Proof Suppose two ground programs B_1 and B_2 which are not strongly equivalent. Put $O = \{L\}$ such that L appears nowhere in B_1, but B_2 contains the constraint $\leftarrow L$. With this setting, for some answer set S of B_1, R_S includes a rule r such that $head(r) = L$, $body^+(r) \subseteq S$ and $body^-(r) \cap S = \emptyset$. Also, for some answer set T of B_2, R_T includes a rule r such that $head(r) = L$, $body^+(r) \subseteq T$ and $body^-(r) \cap T = \emptyset$. In this case, however, $B_1 \cup R_S$ is consistent, but $B_2 \cup R_T$ is inconsistent. Hence, B_1 and B_2 are not inductively equivalent. The converse implication clearly holds. □

6 Discussion

6.1 Comparison of Conditions for Inductive Equivalence

The results of this paper are summarized in Table 2. When the representation language is clausal logic, logical equivalence is necessary and sufficient for inductive equivalence between two background theories in each induction under both classical and the minimal model semantics. By contrast, when the representation language is nonmonotonic logic programming, strong equivalence is necessary and sufficient for inductive equivalence between two background theories in each induction under the answer set semantics. Since $B_1 \equiv B_2$ iff $B_1 \equiv_s B_2$ in clausal logic under both $SEM(B) = Mod(B)$

Table 2 Comparison of Conditions for Inductive Equivalence

	Representation language (semantics)	
Induction	Clausal logic (Mod, MM)	Nonmonotonic LP (AS)
CauInd	$B_1 \equiv B_2$	$B_1 \equiv_s B_2$
BraInd	$B_1 \equiv B_2$	$B_1 \equiv_s B_2$
LFS	$B_1 \equiv B_2$	$B_1 \equiv_s B_2$
DesInd	$B_1 \equiv B_2$	$B_1 \equiv_s B_2$

and $SEM(B) = MM(B)$, we can conclude that strong equivalence of two background theories is necessary and sufficient for inductive equivalence in each induction. On the other hand, the condition of strong equivalence is sometimes relaxed to weak equivalence or other weaker equivalence relations in particular induction algorithms under restricted problem settings.

From the computational viewpoint, testing strong equivalence of propositional EDPs is converted to the problem of propositional entailment in classical logic (Lin 2002). The problem of testing strong equivalence is then solved using existing SAT solvers. For predicate programs with a finite domain, testing strong equivalence is also possible by instantiating a program into a finite propositional one. There is a system for testing strong equivalence of function-free finite nonmonotonic logic programs (Janhunen and Oikarinen 2004). Existence of no procedure for testing strong equivalence of logic programs with functions would restrict practical application of inductive equivalence in ILP. Nevertheless, inductive equivalence is efficiently testable when background theories are given as function-free finite Horn logic program (or Datalog) or a database that is a collection of propositional sentences.

6.2 Relation to Abductive Equivalence

Inoue and Sakama (2005, 2006a,b) have studied equivalence relations in abductive frameworks. Given a background theory B and a set A of candidate hypotheses (called *abducibles*), an abductive framework is defined as a tuple $\langle B, A \rangle$. Two abductive frameworks $\langle B_1, A_1 \rangle$ and $\langle B_2, A_2 \rangle$ are called *explainable equivalent* if, for any observation O, there is an explanation of O in $\langle B_1, A_1 \rangle$ iff there is an explanation of O in $\langle B_2, A_2 \rangle$. On the other hand, two programs are called *explanatorily equivalent* if, for any observation O, O is an explanation of O in $\langle B_1, A_1 \rangle$ iff O is an explanation of O in $\langle B_2, A_2 \rangle$. The former compares explainability of observations in different background theories, while the latter compares explanation contents of observations. Explanatory equivalence is stronger than explainable equivalence, and the former implies the latter. The paper (Inoue and Sakama 2005) introduces two equivalence notions for first-order abduction and abductive logic programming (ALP), and the paper (Inoue and Sakama 2006a) applies the notion to *extended abduction* of (Inoue and Sakama 1995). The paper (Inoue and Sakama 2006b) also argues equivalence between *minimal* explanations.

Comparing (Inoue and Sakama 2005, 2006a,b) with our present work, some interesting connections are observed. When underlying logic is first-order logic, logical equivalence of two theories is a necessary and sufficient condition for explanatory equivalence in abduction. When a background theory is represented by a nonmonotonic logic program, on the other hand, $\langle B_1, A_1 \rangle$ and $\langle B_2, A_2 \rangle$ are explanatorily equivalent iff B_1 and B_2 are strongly equivalent. Those results have connection to the results of

Theorems 3.3, 3.7, and 5.4 of this paper. However, there are some important differences between the previous studies on abductive equivalence and the results of this paper. First, the framework of ALP in (Inoue and Sakama 2005, 2006a,b) characterizes equivalence relations in *brave abduction*. That is, given a logic program B, a hypothesis H explains an observation G if G is true in an answer set of $B \cup H$. This paper characterizes the problem of inductive equivalence not only in brave induction, but also in other forms of induction. Second, in abductive frameworks a hypothesis space A is prespecified as abducibles and possible explanations for a given observation are constructed as a subset of abducibles. The existence of A in abductive logic programs results in characterization by *relative* strong equivalence, i.e., two programs B_1 and B_2 are explanatory equivalent iff they are strongly equivalent *with respect to* A. Moreover, in abductive logic programming, abducibles and observations are usually restricted to (ground) literals. In ILP, on the other hand, hypotheses and observations are general rules rather than facts. Besides these differences, both abduction and induction require strong equivalence of two (nonmonotonic) logic programs to identify the results of abductive/inductive inference. The essence of this lies in the fact that abduction and induction are both *ampliative* reasoning and extend theories. Strong equivalence takes the influence of addition of a rule set to each program into account, so that it succeeds in characterizing the effect of abduction/induction that are not captured by weak equivalence of programs. In (Lifschitz et al. 2001), it is argued that strong equivalence is useful to simplify a part of a program without looking at the other parts. On the other hand, a series of studies (Inoue and Sakama 2005, 2006a,b) and the result of this paper reveal that strong equivalence has another important applications for testing equivalence of background theories in abductive and inductive logic programming.

6.3 Program Development in ILP

As presented in Section 4, there are many ILP systems which handle Horn logic programs as background theories. In Horn logic programs, program transformations which preserve weak equivalence of programs are popularly used for optimizing programs. *Partial evaluation* or *unfold/fold* transformations are of this kind (Tamaki and Sato 1984; Pettorossi and Proietti 1994). For instance, given the program

$$B_1 = \{ p(x) \leftarrow q(x), \quad q(x) \leftarrow r(x), \quad r(a) \leftarrow \},$$

unfolding the first clause by the second one results in the program

$$B_2 = \{ p(x) \leftarrow r(x), \quad q(x) \leftarrow r(x), \quad r(a) \leftarrow \}.$$

On the other hand, in B_2 folding the first clause by the second one results in the program B_1. B_1 and B_2 have the same least model thereby weakly equivalent, but not logically equivalent. In ILP, unfolding is often used as an operator for specialization (Boström and Idestam-Almquist 1994), and folding is used as an operator for generalization under the name of *inverse resolution* (Muggleton and Buntine 1992).

In Section 3.2 we observe that logical equivalence of two clausal theories is necessary and sufficient to guarantee inductive equivalence under the minimal model semantics. Since weak equivalence provides a weaker condition than logical equivalence (Proposition 2.3), the condition of weak equivalence of two clausal theories is not sufficient for preserving inductive equivalence under the minimal model semantics in general.

This result brings an important implication in program development in ILP that basic program transformations, such as unfold/fold transformations, are *not* applicable for optimizing background theories in ILP. If used, those transformations change solutions of induction in general. In the above example, B_1 and B_2 are not inductively equivalent in explanatory induction as $H = \{ q(a) \leftarrow \}$ explains $p(a)$ in B_1 but does not in B_2. Nevertheless, those transformations are still effective as far as one uses induction algorithms that require the condition of weak equivalence. In Section 4, we observe that FOIL and GOLEM are of this kind, but PROGOL and CLAUDIEN are not. It is also known that unfold/fold transformations do not preserve strong equivalence of nonmonotonic logic programs (Osorio et al. 2001), so that those transformations cannot be used for program optimization in nonmonotonic ILP without changing solutions in general.

6.4 Verification of Algorithms

If an induction algorithm produces different hypotheses from two different background theories, those theories are considered to be inductively inequivalent. It may happen, however, that some algorithm may produce different hypotheses from two background theories due to its incompleteness/incorrectness. If two strongly equivalent programs induce different hypotheses in face of some observation, it indicates that the induction algorithm is incomplete or incorrect. In this way, inductive equivalence would be used for testing correctness/completeness of induction algorithms. We consider that any induction algorithm should compute the same hypotheses from two different background theories as far as they are inductively equivalent. With this regard, inductive equivalence has an application to verification of induction algorithms.

For another application, inductive equivalence would be used for comparing capabilities of different induction algorithms. Let $\alpha(B, O)$ be the set of hypotheses induced by an algorithm α using a background theory B and an observation O. For two different induction algorithms α_1 and α_2 under a common problem setting, suppose that $\alpha_1(B_1, O) = \alpha_1(B_2, O)$ implies $\alpha_2(B_1, O) = \alpha_2(B_2, O)$, but not vice versa. In this case, α_1 is considered inductively more *sensitive* than α_2 in the sense that α_1 may distinguish different background theories that are not distinguished by α_2. For instance, suppose any ground Horn logic program B and any set O of ground atoms. In this problem setting, we can say that PROGOL is inductively more sensitive than GOLEM, since $bot(B_1, E) = bot(B_2, O)$ implies $rlgs(M_{B_1}, O) = rlgs(M_{B_2}, O)$ but not vice versa. (This is due to the fact that $B_1 \equiv B_2$ implies $B_1 \equiv_w B_2$ but not vice versa.) Thus, inductive equivalence is also useful for evaluating capabilities of induction algorithms.

7 Concluding Remarks

This paper has studied equivalence issues in induction and inductive logic programming. We introduced the notion of inductive equivalence which compares hypotheses that explain observations with respect to different background theories. Two different logics for representation languages – clausal theories and nonmonotonic logic programming, and four different frameworks of induction – cautious induction, brave induction, learning from satisfiability, and descriptive induction, were considered. The results of this paper show that logical equivalence is necessary and sufficient for inductive equivalence in clausal theories, while strong equivalence is necessary and sufficient in non-

monotonic extended disjunctive programs. On the other hand, we also observed that existing Horn ILP systems sometimes require weaker conditions of equivalence under restricted problem settings. We addressed that inductive equivalence has potential applications for verification and evaluation of induction algorithms. We also argued that program transformations which are popularly used in logic programming generally do not preserve inductive equivalence of programs. This is an important caution for program development in ILP which has been receiving little attention in the field.

Inductive equivalence considered in this paper guarantees coincidence of every hypothesis induced by different background theories. In practice, however, the exact coincidence of whole hypotheses is not always requested and one may be interested in preserving some *preferred* hypotheses. The criteria of preference of hypothesis depends on applications and it is often specified under the name of *induction bias*. In the context of abduction, preferred hypotheses are referred to "best explanations". In (Inoue and Sakama 2006b), it is proved that two abductive theories are explanatory equivalent iff they have the same *minimal explanations* for any observation. Sakama and Inoue (1995) introduce several program transformations which preserve best explanations in abductive logic programming. Inductive equivalence of preferred hypotheses and program transformations for preserving those hypotheses are left for future research.

References

Badea, L. and Stanciu, M. (1999). Refinement operators can be (weakly) perfect. In: *Proceedings of the 9th International Workshop on Inductive Logic Programming*, Lecture Notes in Artificial Intelligence, vol. 1634, pp. 21–32, Springer.

Baral, C. and Gelfond, M. (1994). Logic programming and knowledge representation. *Journal of Logic Programming*, 19/20, 73–148.

Bossu, G. and Siegel, P. (1985). Saturation, nonmonotonic reasoning and the closed-world assumption. *Artificial Intelligence*, 25, 13–63.

Boström, H. and Idestam-Almquist, P. (1994). Specialization of logic programs by pruning SLD-trees. In: *Proceedings of the 4th International Workshop on Inductive Logic Programming*, pp. 31–48.

Clark, K. L. (1978). Negation as failure. In: H. Gallaire and J. Minker (Eds.), *Logic and Data Bases* (pp. 293–322), Plenum, New York.

De Raedt, L. and Bruynooghe, M. (1993). A theory of clausal discovery. In: *Proceedings of the 13th International Joint Conference on Artificial Intelligence*, pp. 1058–1063, Morgan Kaufmann.

De Raedt, L. and Lavrač, N. (1993). The many faces of inductive logic programming. In: *Methodologies for Intelligent Systems, 7th International Symposium*, Lecture Notes in Computer Science, vol. 689, 435–449, Springer.

De Raedt, L. (1997). Logical settings for concept-learning. *Artificial Intelligence*, 95, 187–201.

De Raedt, L. and Dehaspe, L. (1997a). Learning from satisfiability. In: *Proceedings of the 9th Dutch Conference on Artificial Intelligence* (pp. 303–312).

De Raedt, L. and Dehaspe, L. (199b). Clausal discovery. *Machine Learning*, 26(2-3), 99–146.

Denecker, M. and Kakas, A. C. (2002). Abductive logic programming. In: A. C. Kakas and F. Sadri (eds.), *Computational Logic: Logic Programming and Beyond—Essays in Honour of Robert A. Kowalski, Part I*, Lecture Notes in Artificial Intelligence, vol. 2407, pp. 402–436, Springer.

Eiter, T. and Fink, M. (2003). Uniform equivalence of logic programs under the stable model semantics. In: *Proceedings of the 19th International Conference on Logic Programming*, Lecture Notes in Computer Sciences, vol. 2916, pp. 224–238, Springer.

Flach, P. A. (1996). Rationality postulates for induction. In: *Proceedings of the 6th International Conference on Theoretical Aspects of Rationality and Knowledge* (pp. 267–281), Morgan Kaufmann.

Flach, P. A. and Kakas, A. C. (2000). Abductive and inductive reasoning: background and issues. In: P. A. Flach and A. C. Kakas (Eds.). *Abduction and Induction — Essays on their Relation and Integration* (pp. 1–27), Kluwer Academic.

Gelfond, M. and Lifschitz, V. (1988). The stable model semantics for logic programming. In: *Proceedings of the 5th International Conference and Symposium on Logic Programming*, pp. 1070–1080, MIT Press.

Gelfond, M. and Lifschitz, V. (1991). Classical negation in logic programs and disjunctive databases. *New Generation Computing*, 9, 365–385.

Inoue, K. and Sakama, C. (1995). Abductive framework for nonmonotonic theory change. In: *Proceedings of the 14th International Joint Conference on Artificial Intelligence*, pp. 204–210, Morgan Kaufmann.

Inoue, K. (2004). Induction as consequent finding. *Machine Learning*, 55, 109–135.

Inoue, K. and Sakama, C. (2004). Equivalence of logic programs under updates. In: *Proceedings of the 9th European Conference on Logics in Artificial Intelligence*, Lecture Notes in Artificial Intelligence, vol. 3229, pp. 174–186, Springer.

Inoue, K. and Sakama, C. (2005). Equivalence in abductive logic. In: *Proceedings of the 19th International Joint Conference on Artificial Intelligence*, pp. 472–477.

Inoue, K. and Sakama, C. (2006). On abductive equivalence. In: Lorenzo Magnani (ed.), *Model-Based Reasoning in Science and Engineering: Cognitive Science, Epistemology, Logic. Studies in Logic*, pp. 333–352, College Publications, London.

Inoue, K. and Sakama, C. (2006). Abductive equivalence in First-Order Logic. *Logic Journal of the IGPL. Special Issue: Abduction, Practical Reasoning, and Creative Inferences in Science*, 14(2), 333-346.

Janhunen, T. and Oikarinen, E. (2004). LPEQ and DLPEQ – translators for automated equivalence testing of logic programs. In: *Proceedings of the 7th International Conference of Logic Programming and Nonmonotonic Reasoning*, Lecture Notes in Artificial Intelligence, vol. 2923, pp. 336–340, Springer.

Lachiche, N. (2000). Abduction and induction from a non-monotonic reasoning perspective. In: P. A. Flach and A. C. Kakas (Eds.). *Abduction and Induction — Essays on their Relation and Integration* (pp. 107–116), Kluwer Academic.

Lifschitz, V., Pearce, D. and Valverde, A. (2001). Strongly equivalent logic programs. *ACM Transactions on Computational Logic*, 2, 526–541.

Lin, F. (2002). Reducing strong equivalence of logic programs to entailment in classical propositional logic. In: *Proceedings of the 8th International Conference on Principles of Knowledge Representation and Reasoning*, pp. 170–176, Morgan Kaufmann.

Maher, M. J. (1988). Equivalence of logic programs. In: J. Minker (ed.), *Foundations of Deductive Databases and Logic Programming*, pp. 627–658, Morgan Kaufmann.

McCarthy, J. (1980). Circumscription – a form of nonmonotonic reasoning, *Artificial Intelligence*, 13, 27–39.

Minker, J. (1982). On indefinite data bases and the closed world assumption. In: *Proceedings of the 6th International Conference on Automated Deduction*, Lecture Notes in Computer Science, vol. 138 (pp. 292–308), Springer.

Muggleton, S. and Feng, C. (1992). Efficient induction algorithm. In: *Inductive Logic Programming* (S. Muggleton ed.), Academic Press, pp. 281–298.

Muggleton, S. (ed.) (1992). *Inductive Logic Programming*, Academic Press.

Muggleton, S. and Buntine, W. (1992). Machine invention of first-order predicate by inverting resolution. In: *Inductive Logic Programming* (S. Muggleton ed.), Academic Press, pp. 261–280.

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

Nienhuys-Cheng, S.-H. and De Wolf, R. (1997). *Foundations of inductive logic programming*, Lecture Notes in Artificial Intelligence, vol. 1228, Springer.

Otero, R. P. (2001). Induction of stable models. In: *Proceedings of the 11th International Conference on Inductive Logic Programming*, Lecture Notes in Artificial Intelligence, vol. 2157, 193–205, Springer.

Osorio, M., Navarro, J. A. and Arrazola, J. (2001). Equivalence in answer set programming. In: *Proceedings of the 11th International Workshop on Logic Based Program Synthesis and Transformation*, Lecture Notes in Computer Science, vol. 2372, pp. 57–75, Springer.

Pettorossi, A. and Proietti, M. (1994). Transformation of logic programs: foundations and techniques. *Journal of Logic Programming*, 19/20, 261–320.

Plotkin, G. D. (1971). A further note on inductive generalization. In: B. Meltzer and D. Michie (Eds.), *Machine Intelligence*, vol. 6 (pp. 101–124), Edinburgh University Press.

Quinlan, R. (1990). Learning logical definitions from relations. *Machine Learning*, 5, 239–266.

Sagiv, Y. (1988). Optimizing Datalog programs. In: J. Minker (ed.), *Foundations of Deductive Databases and Logic Programming*, pp. 659–668, Morgan Kaufmann.

Sakama, C. and Inoue, K. (1995). The effect of partial deduction in abductive reasoning. In: *Proceedings of the 12th International Conference on Logic Programming*, pp. 383–397, MIT Press.

Sakama, C. and Inoue, K. (2005). Inductive equivalence of logic programs. In: *Proceedings of the 15th International Conference on Inductive Logic Programming*, Lecture Notes in Artificial Intelligence, vol. 3625, pp. 312–329, Springer.

Sakama, C. and Inoue, K. (2009a). Equivalence issues in abduction and induction. *Journal of Applied Logic*, 7(3):318-328.

Sakama, C. and Inoue, K. (2009b). Brave induction: a logical framework for learning from incomplete information. *Machine Learning*, 76(1), 3–35.

Tamaki, H. and Sato, T. (1984). Unfold/Fold transformation of logic programs. In: *Proceedings of the 2nd International Conference on Logic Programming*, pp. 127–138.

Turner, H. (2003). Strong equivalence made easy: nested expressions and weight constraints. *Theory and Practice of Logic Programming*, 3(4–5), 609–622.

Van Emden, M. H. and Kowalski, R. A. (1976). The semantics of predicate logic as a programming language. *Journal of the ACM* 23(4), 733–742.