Inverse Entailment for Full Clausal Theories

Katsumi Inoue
Department of Electrical and Electronics Engineering
Kobe University
Rokkodai, Nada, Kobe 657-8501, Japan
`inoue@eedept.kobe-u.ac.jp`

May 30, 2001

Abstract

This paper shows a sound and complete method for inverse entailment in inductive logic programming. We show that inverse entailment can be computed with a resolution method for consequence-finding. In comparison with previous work, induction via consequence-finding is sound and complete for finding hypotheses from full clausal theories, and can be used for inducing not only definite clauses but also non-Horn clauses and integrity constraints.

1 Introduction

Both *induction* and *abduction* are ampliative reasoning, and agree with the logic to seek hypotheses to account for given observations and examples. That is, given a background theory B and observations (or positive examples) E, the task of induction and abduction is common in finding hypotheses H such that

$$B \wedge H \models E, \tag{1}$$

where $B \wedge H$ is consistent [7, 3, 6, 10]. While the logic is in common, they differ in the usage in applications. According to Peirce, abduction infers a cause of an observation, and can infer something quite different from what is observed. On the other hand, induction infers something to be true through generalization of a number of cases of which the same thing is true. The relation, difference, similarity, and interaction between abduction and induction are extensively studied by authors in [4].

Compared with automated abduction, one of the major drawbacks of automated induction is that computation of inductive hypotheses require a large amount of search space that is highly expensive. General mechanisms to construct hypotheses rely on *refinement* of current hypotheses, which has a lot of alternative choices unless good heuristics is incorporated into search. We thus need a logically principled way to compute inductive hypotheses. One such a promising method to compute hypotheses H in (1) is based on *inverse entailment*, which transforms the equation (1) into

$$B \wedge \neg E \models \neg H. \tag{2}$$

The equation (2) says that, given B and E, any hypothesis H *deductively* follows from $B \wedge \neg E$ in its negated form. For example, given $B_1 = \{human(s)\}$ and $E_1 = \{mortal(s)\}$,

$$H_1 = \{\forall x(human(x) \supset mortal(x))\}$$

satisfies (1). In fact,

$$B_1 \wedge \neg E_1 = \{human(s), \neg mortal(s)\} \models \exists x(human(x) \wedge \neg mortal(x)) = \neg H_1.$$

The equation (2) can be seen in literature, e.g., [8] for abduction and [13] for induction.

While the equation (2) is useful for computing abductive explanations of observations in abduction, it is more difficult to apply it to compute inductive hypotheses. In abduction, without loss of generality, E is written in a ground atom, and each H is usually assumed to be a conjunction of literals. These conditions make abductive computation relatively easy, and *consequence-finding* algorithms [8, 2, 12] can be directly applied.

In induction, however, E can be clauses and H is usually a general rule. Universally quantified rules for H cannot be easily obtained from the negation of consequences of $B \wedge \neg E$. Then, Muggleton [13] considered the so called *bottom clause* $\bot(E, B)$:

$$\bot(E, B) = \{\neg L \mid L \text{ is a literal and } B \wedge \neg E \models L\}.$$

A hypothesis H is then constructed by generalizing a sub-clause of $\bot(E, B)$, i.e.,

$$H \models \bot(E, B).$$

While this method is adopted in Progol, it is incomplete for finding hypotheses satisfying (1). Then, several improvements have been reported to make inverse entailment complete [14, 5, 19] or to characterize inverse entailment precisely [17, 18, 15]. However, such improved inductive procedures are not very simple when compared with abductive computation. More seriously, some improved procedures are unsound even though they are complete. Another difficulty in the previous inductive methods lies in the facts: (i) each constructed hypothesis in H is usually assumed to be a Horn clause, (ii) the example E is given as a single Horn clause, and (iii) the background theory B is a set of Horn clauses. Finding full clausal hypotheses from full clausal theories have not been paid much attention so far.

In this paper, we propose a simple, yet powerful method to handle inverse entailment (2) for computing inductive hypotheses. Unlike previous methods based on the bottom clause, we do not restrict the consequences of $B \wedge \neg E$ to literals, but consider the *characteristic clauses* of $B \wedge \neg E$, which were originally proposed for AI applications (including abduction) of consequence-finding [8]. Using our method, sound and complete hypothesis-finding from full clausal theories can be realized, and not only definite clauses but also non-Horn clauses and integrity constraints can be constructed as H. By this way, inductive algorithms can be designed with deductive procedures, which reduce search space as much as possible like computing abduction.

This paper is organized as follows. Section 2 introduces the theoretical background in this paper. Section 3 provides the basic idea called *CF-induction* to construct inductive hypotheses using a consequence-finding method. Section 4 discusses related work, and Section 5 is the conclusion. The proof of the main theorem is given in the appendix. Due to the lack of space, we omit some properties of CF-induction as well as their proofs in this paper. In the full paper, all proofs will be addressed. Moreover, induction and abduction will be compared from the viewpoint of consequence-finding, and the relationship and difference between the two will also be clarified in the full paper.

2 Background

2.1 Inductive Logic Programming

Here, we review the terminology of inductive logic programming (ILP). A *clause* is a disjunction of literals, and is often denoted as the set of its disjuncts. A clause $\{A_1, \dots, A_m, \neg B_1, \dots, \neg B_n\}$, where each A_i, B_j is an atom, is also written as $B_1 \wedge \cdots \wedge B_n \supset A_1 \vee \cdots \vee A_m$. Any variable in a clause is assumed to be universally quantified at the front. A *definite clause* is a clause which contains only one positive literal. A *positive* (*negative*) *clause* is a clause whose disjuncts are all positive (negative) literals. A negative clause is often called an *integrity constraint*. A *Horn clause* is a definite clause or negative clause; otherwise it is *non-Horn*. A *clausal theory* Σ is a finite set of clauses. A clausal theory is *full* if it contains non-Horn clauses.

A (*universal*) *conjunctive normal form* (CNF) formula is a conjunction of clauses, and a *disjunctive normal form* (DNF) formula is a disjunction of conjunctions of literals. A clausal theory Σ is identified with the CNF formula that is the conjunction of all clauses in Σ. We define the *complement* of a clausal theory, $\Sigma = C_1 \wedge \cdots \wedge C_k$ where each C_i is a clause, as the DNF formula $\neg C_1\sigma_1 \vee \cdots \vee \neg C_k\sigma_k$, where $\neg C_i = B_1 \wedge \cdots \wedge B_n \wedge \neg A_1 \wedge \cdots \wedge \neg A_m$ for $C_i = (B_1 \wedge \cdots \wedge B_n \supset A_1 \vee \cdots \vee A_m)$, and σ_i is a substitution which replaces each variable x in C_i with a Skolem constant sk_x. This replacement of variables reflects the fact that each variable in $\neg C_i$ is existentially quantified at the front. Since there is no ambiguity, we write the complement of Σ as $\neg\Sigma$.

Let C and D be two clauses. C *subsumes* D if there is a substitution θ such that $C\theta \subseteq D$. When C subsumes D, we also say that C is a *generalization* of D. C *properly subsumes* D if C subsumes D but D does not subsume C. For a clausal theory Σ, $\mu\Sigma$ denotes the set of clauses in Σ not properly subsumed by any clause in Σ.

Let B, E, and H are clausal theories, representing a *background theory*, *(positive) examples*, and *hypotheses*, respectively. The most popular formalization of concept-learning is *learning from entailment* (or *explanatory induction*), in which the task is: given B and E, find H such that $B \wedge H \models E$ and $B \wedge H$ is consistent. Note that we do not consider *negative examples* in this paper.

2.2 Consequence-Finding

For a clausal theory Σ, a *consequence* of Σ is a clause entailed by Σ. We denote by $Th(\Sigma)$ the set of all consequences of Σ. The *consequence-finding* problem is firstly addressed by Lee [11] in the context of resolution principle. Lee proved that, for any consequence D of Σ, the resolution principle can derive a clause C from Σ such that C entails D. In this sense, the resolution principle is said *complete for consequence-finding*. In Lee's theorem, "C entails D" can be replaced with "C subsumes D". Hence, the consequences of Σ that are derived by the resolution principle includes $\mu Th(\Sigma)$. The notion of consequence-finding is used as the theoretical background for discussing the completeness of ILP systems [16].

By extending the notion of consequence-finding, Inoue [8] defined *characteristic clauses* to represent "interesting" clauses for a given problem. Each characteristic clause is constructed over a sub-vocabulary of the representation language called a *production field*. In this paper, for the sake of simplicity, a production field $\mathcal{P}$ is defined as a set of distinguished literals. Let $Th_{\mathcal{P}}(\Sigma)$ be the clauses in $Th(\Sigma)$ all of whose literals belong to $\mathcal{P}$. Then, the *characteristic clauses* of Σ with respect to $\mathcal{P}$ are defined as:

$$Carc(\Sigma, \mathcal{P}) = \mu\, Th_{\mathcal{P}}(\Sigma)\,.$$

Here, we do not include any tautology $\neg L \vee L$ ($\equiv True$) in $Carc(\Sigma, \mathcal{P})$ even when both L and $\neg L$ belong to $\mathcal{P}$. Note that the empty clause $\Box$ is the unique clause in $Carc(\Sigma, \mathcal{P})$ if and only if Σ is unsatisfiable. This means that proof-finding is a special case of consequence-finding. In the propositional case, each characteristic clause of Σ is a *prime implicate* of Σ.

When a new clause C is added to a clausal theory Σ, some consequences are newly derived with this new information. Such a new and "interesting" clause is called a "new" characteristic clauses. Formally, the *new characteristic clauses* of C with respect to Σ and $\mathcal{P}$ are:

$$\begin{aligned} NewCarc(\Sigma, C, \mathcal{P}) &= \mu\,[\, Th_{\mathcal{P}}(\Sigma \wedge C) - Th(\Sigma)\,] \\ &= Carc(\Sigma \wedge C, \mathcal{P}) - Carc(\Sigma, \mathcal{P}). \end{aligned}$$

When a new formula is not a single clause but a CNF formula $F = C_1 \wedge \cdots \wedge C_m$, where each C_i is a clause, $\mathit{NewCarc}(\Sigma, F, \mathcal{P})$ can be decomposed into m *NewCarc* operations each of whose added new formula is a single clause [8]:

$$\mathit{NewCarc}(\Sigma, F, \mathcal{P}) = \mu\,[\, \bigwedge_{i=1}^{m} \mathit{NewCarc}(\Sigma_i, C_i, \mathcal{P})\,]\,, \tag{3}$$

where $\Sigma_1 = \Sigma$, and $\Sigma_{i+1} = \Sigma_i \wedge C_i$, for $i = 1, \ldots, m-1$. This incremental computation can be applied to get the characteristic clauses of Σ with respect to $\mathcal{P}$ as follows.

$$\mathit{Carc}(\Sigma, \mathcal{P}) = \mathit{NewCarc}(\mathit{True}, \Sigma, \mathcal{P}). \tag{4}$$

Several procedures have been proposed to compute (new) characteristic clauses. For example, *SOL resolution* [8] is an extension of the Model Elimination (ME) calculus to which the Skip rule is introduced. In computing $\mathit{NewCarc}(\Sigma, C, \mathcal{P})$, SOL resolution treats a newly added clause C as the *top clause* input to ME, and derives those consequences relevant to C directly. With the Skip rule, SOL resolution focuses on deriving only those consequences belonging to the production field $\mathcal{P}$. Various pruning methods are also introduced to enhance the efficiency of SOL resolution in a *connection-tableau* format [9]. Instead of ME, *SFK resolution* [2] is a variant of ordered resolution, which is enhanced with the Skip rule for finding characteristic clauses. An extensive survey of consequence-finding algorithms in propositional logic is given by Marquis [12].

3 Induction as Consequence-Finding

In this section, we characterize explanatory induction by consequence-finding. Suppose that we are given a background theory B and examples E, both of which are clausal theories (or CNF) possibly containing non-Horn clauses. Recall that explanatory induction seeks a clausal theory H such that:

$$B \wedge H \models E,$$
$$B \wedge H \text{ is consistent.}$$

These two are equivalent to

$$B \wedge \neg E \models \neg H, \tag{5}$$
$$B \not\models \neg H. \tag{6}$$

Like inverse entailment, we are interested in some formulas derived from $B \wedge \neg E$ that are not derived from B alone. Here, instead of the bottom clause $\bot(B, E)$ in [13], we consider some clausal theory $CC(B, E)$. Then, the equation (5) can be written as

$$B \wedge \neg E \models CC(B, E), \tag{7}$$
$$CC(B, E) \models \neg H. \tag{8}$$

The latter (8) is also written as

$$H \models \neg CC(B, E). \tag{9}$$

Also, by (6) and (8), we have

$$B \not\models CC(B, E). \tag{10}$$

By (7), $CC(B, E)$ is obtained by computing the characteristic clauses of $B \wedge \neg E$ because any other consequence of $B \wedge \neg E$ can be obtained by constructing a clause that is subsumed by a characteristic clause. Hence,

$$Carc(B \wedge \neg E, \mathcal{P}) \models CC(B, E), \tag{11}$$

where the production field $\mathcal{P}$ ($\subseteq \mathcal{L}$) is defined as some set of literals possibly considering an *inductive bias*. Here, $\mathcal{L}$ is the set of all literals in the first-order language. The other requirement for $CC(B, E)$ is the equation (10), which is satisfied if at least one of clauses in $CC(B, E)$ is not a consequence of B; otherwise, $CC(B, E)$ is entailed by B. This is realized by including a clause from $NewCarc(B, \neg E, \mathcal{P})$ in $CC(B, E)$.

In constructing hypotheses H from the clausal theory $CC(B, E)$, notice that $\neg CC(B, E)$ is entailed by H in (9). Since $\neg CC(B, E)$ is DNF, we convert it into the CNF formula F, i.e., $F \equiv \neg CC(B, E)$. Then, H is constructed as a clausal theory which entails F, i.e., $H \models F$. There are several methods to perform such an inverse of entailment in ILP. The most popular one is the use of a *generalizer* (see [19], for example) which constructs a clausal theory H such that every clause in F is *entailed* by some clause in H.[1] In some case, reverse Skolemization in abduction [1] also works as a generalizer, but there exist other techniques to generalize clauses such as anti-instantiation (i.e., replacement of terms with variables), addition of clauses, dropping literals from clauses, and Plotkin's least generalization of multiple clauses. Note that applying arbitrary generalizer to F may cause an inconsistency of H with B. To ensure that $B \wedge H$ is consistent, the clauses of H must keep those literals that are generalizations of the complement of at least one clause from $NewCarc(B, \neg E, \mathcal{P})$.

Now, the whole algorithm to construct inductive hypotheses is as follows.

Definition 3.1 Let B and E be clausal theories. A clausal theory H is derived by a *CF-induction* from B and E if H is constructed as follows.

Step 1. Compute $Carc(B \wedge \neg E, \mathcal{P})$;[2]

Step 2. Construct $CC(B, E) = C_1 \wedge \cdots \wedge C_m$, where each C_i is a clause satisfying the conditions:

(a) Each C_i is a variant or an instance of a clause from $Carc(B \wedge \neg E, \mathcal{P})$;

(b) At least one C_i is a variant or an instance of a clause from $NewCarc(B, \neg E, \mathcal{P})$;

Step 3. Convert $\neg CC(B, E)$ into the CNF formula F;

Step 4. H is obtained by applying a generalizer to F under the constraint that $B \wedge H$ is consistent.

Example 3.1 The following theory is often used to illustrate how the bottom clause is used in inverse entailment [17, 18]. Consider

$$\begin{aligned} B_2 &= (cat(x) \supset pet(x)) \wedge \\ &\quad (small(x) \wedge fluffy(x) \wedge pet(x) \supset cuddly\_pet(x)), \\ E_2 &= (fluffy(x) \wedge cat(x) \supset cuddly\_pet(x)). \end{aligned}$$

Then, the complement of E_2 is

$$\neg E_2 = fluffy(sk_x) \wedge cat(sk_x) \wedge \neg cuddly\_pet(sk_x)),$$

[1] If "entailment" is replaced with "subsumption" here, the completeness in Theorem 3.1 does not precisely hold.

[2] Since the number of characteristic clauses may be large or infinite in general, this step should be interleaved on demand with construction of each C_i at Step 2 in practice.

and $NewCarc(B_2, \neg E_2, \mathcal{L})$ is

$$\neg E_2 \wedge pet(sk_x) \wedge \neg small(sk_x).$$

Let $CC(B_2, E_2) = NewCarc(B_2, \neg E_2, \mathcal{L})$. In this case, $F_2 = \neg CC(B_2, E_2) = \bot(B_2, E_2)$ hold. By applying reverse Skolemization to F_2, we get the hypothesis H_2:

$$\mathit{fluffy}(x) \wedge cat(x) \wedge pet(x) \supset cuddly\_pet(x) \vee small(x).$$

While in the above cited reference the subclause

$$\mathit{fluffy}(x) \wedge cat(x) \supset small(x)$$

is adopted for a definite clause, H_2 is the most-specific hypothesis in the sense of [13].

Example 3.2 [17] This example illustrates the incompleteness of inverse entailment based on the bottom clause in [13]. Consider the background theory and the example:

$$\begin{aligned} B_3 &= even(0) \wedge (\neg odd(x) \vee even(s(x))), \\ E_3 &= odd(s(s(s(0)))). \end{aligned}$$

Then, $Carc(B_3 \wedge \neg E_3, \mathcal{L}) = B_3 \wedge \neg E_3$. Suppose that $CC(B_3, E_3)$ is chosen as:

$$even(0) \wedge (\neg odd(s(0)) \vee even(s(s(0)))) \wedge \neg odd(s(s(s(0)))),$$

where the second clause is an instance of the latter clause in B_3, and the third clause belongs to $NewCarc(B_3, \neg E_3, \mathcal{L})$. By converting $\neg CC(B_3, E_3)$ into CNF, F_3 consists of the clauses:

$$\neg even(0) \vee odd(s(0)) \vee odd(s(s(s(0)))),$$
$$\neg even(0) \vee \neg even(s(s((0))) \vee odd(s(s(s(0)))).$$

Considering the single clause:

$$H_3 = \neg even(x) \vee odd(s(x)),$$

H_3 subsumes both clauses in F_3, so is the hypothesis. On the other hand, the bottom clause is

$$\bot(B_3, E_3) = \neg even(0) \vee odd(s(s(s(0)))),$$

from which H_3 cannot be obtained by any generalizer.

Below is the correctness result for clausal theories derived using CF-induction. The result implies not only the completeness but the soundness of CF-induction.

Theorem 3.1 *Let B, E and H be clausal theories. H is derived by a CF-induction from B and E if and only if $B \wedge H \models E$ and $B \wedge H$ is consistent.*

Proof: The proof is given in the appendix. □

In Theorem 3.1, both B and E may contain non-Horn clauses and integrity constraints. Also, the derived hypotheses H may be non-Horn. This result is an answer to the open question posed by [14], as to whether a generalization of inverse entailment would be complete for arbitrary clausal background theories.

4 Related Work

CF-induction is obviously influenced by previous work on inverse entailment (IE), which was initiated by Muggleton [13]. The original IE allows Horn clauses for B and a single Horn clause for each of B and E. Even in this setting, however, the method based on $\bot(B, E)$ is incomplete for finding H such that $B \wedge H \models E$ [17]. Muggleton [14] considers an enlarged bottom set to make IE complete, but the revised method is unsound. Furukawa [5] also proposes a complete algorithm, but it is relatively complex. Yamamoto [18] shows that a variant of SOL resolution can be used to implement IE based on $\bot(B, E)$. However, he computes positive and negative parts in $\bot(B, E)$ separately, where SOL resolution is used only for computing positive literals. Muggleton and Bryant [15] suggest the use of Stickel's PTTP for implementing theory completion using IE, which seems inefficient since PTTP is not a consequence-finding procedure but a theorem prover. Compared with these previous work, CF-induction proposed in this paper is simple, yet sound and complete for finding hypotheses from full clausal theories. Instead of the bottom clause, CF-induction uses the characteristic clauses, which strictly include the literals in $\overline{\bot(B, E)}$.

Yamamoto and Fronhöfer [19] firstly extend IE to allow for full clausal theories for B and E, and introduce the *residue hypothesis* for ground instances of $B \wedge \neg E$. Roughly speaking, a residue hypothesis corresponds to the enumeration of all paths in the matrix in Bibel's Connection method. By contrast, CF-induction is realized by a resolution-based consequence-finding procedure, which naturally extends most previous work on IE, and can easily handle non-ground clauses. Compared with the procedure by [19], a merit of CF-induction is the existence of a *production field* $\mathcal{P}$, which can be used to guide and restrict derivations of clauses by reflecting an *inductive bias*.

5 Conclusion

In this paper, we put emphasis on the completeness of inverse entailment in full clausal theories. To this end, we proposed CF-induction, which is sound and complete for finding hypotheses from full clausal theories. CF-induction performs induction via consequence-finding, which enables us to generate inductive hypotheses in a logically principled way. CF-induction can be implemented with existing systematic consequence-finding procedures such as SOL resolution [8] and SFK resolution [2].

Acknowledgements

The author thanks Akihiro Yamamoto for fruitful discussion on the topic of this paper.

A Proof of Theorem 3.1

We prove the correctness of CF-induction by giving its soundness and completeness. Let B, E and H be clausal theories. Section A.1 shows that for any H derived by a CF-induction from B and E, it holds that $B \wedge H \models E$ and $B \wedge H$ is consistent. Section A.2 shows the converse, that is, if $B \wedge H \models E$ and $B \wedge H$ is consistent then H is derived by a CF-induction from B and E.

In the following, we assume the language $\mathcal{L}_H$ for all hypotheses H's. Usually, $\mathcal{L}_H$ is given as the set of all clauses constructed from the first-order language, but we can restrict the form of hypotheses by considering an inductive bias with a subset of literals/predicates. Then, the production field $\mathcal{P}$ is set to the complement of $\mathcal{L}_H$. When $\mathcal{L}_H$ is the set of all clauses, $\mathcal{P}$ is given as $\mathcal{L}$.

A.1 Soundness of CF-Induction

Let H be a hypothesis obtained by a CF-induction from B and E. Then, by the definition of a CF-induction, there is a DNF formula $CC(B,E) = C_1 \wedge \cdots \wedge C_m$ such that

[a] H is obtained by applying a generalizer to the CNF representation of $\neg CC(B,E)$;

[b] every C_i $(i = 1, \ldots, m)$ is a variant or an instance of a clause from $Carc(B \wedge \neg E, \mathcal{P})$; and

[c] there is a C_j $(1 \leq j \leq m)$ that is a variant or an instance of a clause from $NewCarc(B, \neg E, \mathcal{P})$.

Then, by [b], for any C_i $(i = 1, \ldots, m)$, there is a clause $D_i \in Carc(B \wedge \neg E, \mathcal{P})$ such that $B \wedge \neg E \models D_i$ and $D_i \models C_i$. Obviously, it holds that $B \wedge \neg E \models C_i$. Also, by [c], it holds that $B \not\models C_j$. Hence,

$$B \wedge \neg E \models C_1 \wedge \cdots \wedge C_m,$$
$$B \not\models C_1 \wedge \cdots \wedge C_m.$$

Now, let

$$F \equiv \neg CC(B,E) \equiv \neg C_1 \vee \cdots \vee \neg C_m.$$

Then,

$$B \wedge \neg E \models \neg F,$$
$$B \not\models \neg F,$$

which are equivalent to

$$B \wedge F \models E,$$
$$B \wedge F \text{ is consistent.}$$

Finally, $H \models F$ holds by [a], which implies that

$$B \wedge H \models E.$$

The condition that $B \wedge H$ is consistent is included in Step 4 of a CF-induction. □

A.2 Completeness of CF-Induction

Suppose that

$$B \wedge H \models E,$$
$$B \wedge H \text{ is consistent.}$$

Then,

$$B \wedge \neg E \models \neg H,$$
$$B \not\models \neg H.$$

We first show that $\neg H$ is entailed by $Carc(B \wedge \neg E, \mathcal{P})$. Suppose not, i.e., $Carc(B \wedge \neg E, \mathcal{P}) \not\models \neg H$. Then, H is consistent with $Carc(B \wedge \neg E, \mathcal{P})$. Here, by the definition of the characteristic clauses,

$$Carc(B \wedge \neg E, \mathcal{P}) \subseteq Th(B \wedge \neg E).$$

Also, $Carc(B \wedge \neg E, \mathcal{P})$ belongs to the production field $\mathcal{P}$. On the other hand, H belongs to $\mathcal{L}_H$ that is the complement of $\mathcal{P}$. Hence, H does not belong to $\mathcal{P}$. Then, H is consistent with $B \wedge \neg E$. This contradicts the fact that $B \wedge \neg E \models \neg H$. Therefore,

$$Carc(B \wedge \neg E, \mathcal{P}) \models \neg H.$$

Hence, $Carc(B \wedge \neg E, \mathcal{P}) \wedge H$ is unsatisfiable. Now, there are two ways to prove the completeness of CF-induction.

[A] *Using the Herbrand's theorem,* there is a finite set S of ground instances of clauses from $Carc(B \wedge \neg E, \mathcal{P})$ such that $S \wedge H$ is unsatisfiable. This set S can actually be constructed by a CF-induction. In fact, we can set S as $CC(B, E)$. In other words, let us construct

$$CC(B, E) = C_1 \wedge \cdots \wedge C_m$$

at Step 2 of a CF-induction such that

(a) each C_i $(i = 1, \ldots, m)$ is a ground instance of a clause from $Carc(B \wedge \neg E, \mathcal{P})$, and

(b) $C_1 \wedge \cdots \wedge C_m \wedge H$ is unsatisfiable.

Then, there is a C_j $(1 \leq j \leq m)$ that is a ground instance of a clause from $NewCarc(B, \neg E, \mathcal{P})$ (for this, see the discussion below the equation (11) in Section 3). Finally, at Steps 3 and 4, H can be obtained by applying a generalizer (including anti-instantiation) to the CNF representation of $\neg CC(B, E)$. $\square$

[B] *Using the compactness theorem,* there is a finite subset S of $Carc(B \wedge \neg E, \mathcal{P})$ such that $S \wedge H$ is unsatisfiable. In this case, S can also be constructed at Step 2 of a CF-induction as $CC(B, E) = C_1 \wedge \cdots \wedge C_m$, where

(a) every C_i $(i = 1, \ldots, m)$ is a variant of a clause from $Carc(B \wedge \neg E, \mathcal{P})$, and

(b) $C_1 \wedge \cdots \wedge C_m \wedge H$ is unsatisfiable.

Then, there is a C_j $(1 \leq j \leq m)$ that is a variant of a clause in $NewCarc(B, \neg E, \mathcal{P})$ as in the proof of [A]. In this case, however, we have to take care of variables in C_i's. Taking the complement of a C_i, each variable x in C_i becomes a Skolem constant sk_x in $\neg C_i$, in which x is interpreted as existentially quantified. Sometimes we need multiple "copies" of $\neg C_i$ in $\neg CC(B, E)$ using different constants like sk_x^1, sk_x^2, etc, depending on how many times C_i is used to derive $\neg H$ from $B \wedge \neg E$.[3] Then, at Steps 3 and 4, H can be obtained by applying a generalizer to the CNF representation of $\neg CC(B, E)$. $\square$

Example A.1 We now verify the completeness proof [B] by applying it to the theory in Example 3.2, while the proof [A] can easily be checked by following the way shown in Example 3.2. This time, we choose a non-ground characteristic clause as

$$CC'(B_3, E_3) \;=\; even(0) \wedge (\neg odd(x) \vee even(s(x))) \wedge \neg odd(s(s(s(0)))).$$

Then, the complement of $CC'(B_3, E_3)$ becomes

$$\begin{aligned} & (\neg even(0) \vee odd(sk_x) \vee odd(s(s(s(0))))) \\ \wedge \quad & (\neg even(0) \vee \neg even(s(sk_x)) \vee odd(s(s(s(0))))). \end{aligned}$$

Here, we need only one copy of $\neg \forall x(\neg odd(x) \vee even(s(x))) = \exists x(odd(x) \wedge \neg even(x))$. The hypothesis $H_3 = \neg even(x) \vee odd(s(x))$ entails $\neg CC'(B_3, E_3)$ because

$$\forall x(\neg even(x) \vee odd(s(x))) \models \exists y[(\neg even(0) \vee odd(y)) \wedge (\neg even(s(y)) \vee odd(s(s(s(0)))))].$$

To see this, take the substitution $y/s(0)$ in the right side. Then, H_3 subsumes both $\neg even(0) \vee odd(s(0))$ and $\neg even(s(s(0))) \vee odd(s(s(s(0))))$, and thus entails $\neg CC'(B_3, E_3)$.

[3]Recall that a first-order formula and its Skolem standard form is equivalent if the formula is unsatisfiable. However, a Skolem standard form of a satisfiable formula does not preserve the equivalence. Here, we need copies of a Skolemized formula to preserve the completeness of consequence-finding.

References

[1] P.T. Cox and T. Pietrzykowski. Causes for events: their computation and applications. In: *Proceedings of the 8th International Conference on Automated Deduction*, LNCS, 230, pages 608–621, Springer, 1986.

[2] Alvaro del Val. A new method for consequence finding and compilation in restricted languages. In: *Proceedings of AAAI-99*, pages 259–264, AAAI Press, 1999.

[3] Yannis Dimopoulos and Antonis Kakas. Abduction and inductive learning. In: Luc De Raedt, editor, *Advances in Inductive Logic Programming*, pages 144–171, IOS Press, 1996.

[4] Peter A. Flach and Antonis C. Kakas, editors. *Abduction and Induction: Essays on their Relation and Integration.* Kluwer, 2000.

[5] Koichi Furukawa. On the completion of the most specific hypothesis computation in inverse entailment for mutual recursion. In: *Proceedings of Discovery Science '98*, LNAI 1532, pages 315–325, Springer, 1998.

[6] Éric Grégoire and Lakhdar Saïs. Inductive reasoning is sometimes deductive. In: *Proceedings of ECAI-96 Workshop on Abductive and Inductive Reasoning*, 1996.

[7] Nicolas Helft. Induction as nonmonotonic inference. In: *Proceedings of KR '89*, pages 149–156, Morgan Kaufmann, 1989.

[8] Katsumi Inoue. Linear resolution for consequence finding. *Artificial Intelligence*, 56:301–353, 1992.

[9] Koji Iwanuma, Katsumi Inoue, and Ken Satoh. Completeness of pruning methods for consequence finding procedure SOL. In: P. Baumgartner and H. Zhang, editors, *Proceedings of the 3rd International Workshop on First-Order Theorem Proving*, pages 89–100, 2000.

[10] Nicolas Lachiche. Abduction and induction from a non-monotonic reasoning perspective. In: [4], pages 107–116.

[11] Char-Tung Lee. A completeness theorem and computer program for finding theorems derivable from given axioms. Ph.D. thesis, Department of Electrical Engineering and Computer Science, University of California, Berkeley, CA, 1967.

[12] Pierre Marquis. Consequence finding algorithms. In: D. Gabbay and P. Smets, editors, *Handbook for Defeasible Reasoning and Uncertain Management Systems*, Volume 5, Kluwer, 2000.

[13] Stephen Muggleton. Inverse entailment and Progol. *New Generation Computing*, 13:245–286, 1995.

[14] Stephen Muggleton. Completing inverse entailment. In: N. Lavrač and S. Džeroski, editors, *Proceedings of the 7th International Workshop on Inductive Logic Programming*, LNAI 1297, pages 245–249, Springer, 1997.

[15] Stephen Muggleton and Christoper Bryant. Theory completion and inverse entailment. J. Cussens and A. Frisch, editors, *Proceedings of the 10th International Conference on Inductive Logic Programming*, LNAI 1866, pages 130–146, Springer, 2000.

[16] Shan-Hwei Nienhuys-Cheng and Ronald de Wolf. *Foundations of Inductive Logic Programming.* LNAI, 1228, Springer, 1997.

[17] Akihiro Yamamoto. Which hypotheses can be found with inverse entailment? In: N. Lavrač and S. Džeroski, editors, *Proceedings of the 7th International Workshop on Inductive Logic Programming*, LNAI 1297, pages 296–308, Springer, 1997.

[18] Akihiro Yamamoto. Using abduction for induction based on bottom generalization. In: [4], pages 267–280.

[19] Akihiro Yamamoto and Bertram Fronhöfer. Hypotheses finding via residue hypotheses with the resolution principle. In: *Proceedings of the 11th International Conference on Algorithmic Learning Theory*, LNAI 1968, pages 156–165, Springer, 2000.