Machine Learning, 55, 109–135, 2004

Induction as Consequence Finding

KATSUMI INOUE ki@nii.ac.jp
National Institute of Informatics, 2-1-2 Hitotsubashi, Chiyoda-ku, Tokyo 101-8430

Editors: Céline Rouveirol and Michèle Sebag

Abstract. This paper presents a general procedure for inverse entailment which constructs inductive hypotheses in inductive logic programming. Based on inverse entailment, not only unit clauses but also characteristic clauses are deduced from a background theory together with the negation of positive examples. Such clauses can be computed by a resolution method for consequence finding. Unlike previous work on inverse entailment, our proposed method called CF-induction 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. We also show that CF-induction can be used to compute abductive explanations, and then compare induction and abduction from the viewpoint of inverse entailment and consequence finding.

Keywords: induction, abduction, consequence finding, inverse entailment

1. Introduction

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

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

where $B \wedge H$ is consistent (Helft, 1989; Dimopoulos & Kakas, 1996; Grégoire & Saïs, 1996; Lachiche, 2000). While the logic is in common, they differ in the usage in applications. According to Peirce (1932, Paragraph 777), 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 Flach and Kakas (2000).

Compared with automated abduction, one of the major drawbacks of automated induction is that computation of inductive hypotheses requires a large amount of search 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 in 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 Eq. (1) into

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

The Eq. (2) says that, given B and E, any hypothesis H *deductively* follows from $B \wedge \neg E$ in its negated form. For example, suppose that

$$B_1 = human(s), \quad E_1 = mortal(s)$$

are given. Then,

$$H_1 = \forall x\, (human(x) \supset mortal(x))$$

satisfies (1). In fact,

$$human(s) \wedge \neg mortal(s) \models \exists x\, (human(x) \wedge \neg mortal(x)),$$

that is, $B_1 \wedge \neg E_1 \models \neg H_1$. The Eq. (2) is seen in literature, e.g., (Inoue, 1992) for abduction and Muggleton (1995) for induction.

While the Eq. (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 as 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 (Inoue, 1992; del Val, 1999; Marquis, 2000) 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 (1995) introduced a "bridge" formula U between $B \wedge \neg E$ and $\neg H$:

$$B \wedge \neg E \models U, \quad U \models \neg H.$$

As such a bridge formula U, Muggleton considers the conjunction of all unit clauses that are entailed by $B \wedge \neg E$. In this case, $\neg U$ is a clause called the *bottom clause* $\bot(B, E)$. A hypothesis H is then constructed by generalizing a sub-clause of $\bot(B, E)$, i.e., $H \models \bot(B, E)$.

While this method with $\bot(B, E)$ is adopted in Progol (Muggleton, 1995), it is incomplete for finding hypotheses satisfying (1) (Yamamoto, 1997). Then, several improvements have been reported to make inverse entailment complete (Muggleton, 1998; Furukawa, 1998; Yamamoto & Fronhöfer, 2000) or to characterize inverse entailment precisely (Yamamoto, 1997, 2000; Furukawa, 1997; Muggleton & Bryant, 2000). 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 most 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 has not been received 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 (Inoue, 1992). 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. In this way, inductive algorithms can be designed with deductive procedures, which reduce search space as much as possible like in computing abduction. In this paper, we also clarify the relationship and difference between abductive and inductive computation.

This paper is an extended version of Inoue (2001), and contains a variety of new material including complete proofs of all theorems, extensive discussion on inverse entailment, implementation issues, and comparison with related work. This paper is organized as follows. Section 2 introduces the theoretical background in this paper. Section 3 reviews previous approaches to inverse entailment, in which abduction is characterized as a consequence finding method. Section 4 provides the basic idea called *CF-induction* to construct inductive hypotheses using a consequence finding method. Section 5 compares induction with abduction in the context of consequence finding. Section 6 discusses related work, and Section 7 is the conclusion. The proof of the main theorem is given in the appendix.

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 by the set of its disjuncts. A clause $\{A_1, \ldots, A_m, \neg B_1, \ldots, \neg B_n\}$, 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*. The *length* of a clause is the number of literals it contains. A *unit clause* is a clause with the length 1, i.e., a literal. A *clausal theory* Σ is a finite set of clauses. A clausal theory is *full* if it contains non-Horn clauses. On the other hand, a *Horn program* is a clausal theory containing Horn clauses only.

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 $C\theta \subseteq D$. *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 be clausal theories, representing a *background theory*, *(positive) examples*, and a *hypothesis*, 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 here that *negative examples* do not appear in this definition. We will consider negative examples in Section 4.5. On the other hand, in the case of *abduction*, E and H are usually called *observations* and an *explanation*, respectively, for the same task as induction. Precise definitions for abduction and induction are given in Section 3.

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 was first addressed by Lee (1967) in the context of the resolution principle. Lee proved that, for any non-tautological 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 to be *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 $\mu Th(\Sigma)$, and are equivalent under subsumption to the clauses of $\mu Th(\Sigma)$. The notion of consequence finding is used as the theoretical background for discussing the completeness of ILP systems (Nienhuys-Cheng & de Wolf, 1997). In ILP, the completeness result of consequence finding is often called the *subsumption theorem* (Nienhuys-Cheng & de Wolf, 1997).

By extending the notion of consequence finding, Inoue (1992) 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". Formally, a *production field* $\mathcal{P}$ is a pair, $\langle \mathbf{L},\ Cond \rangle$, where $\mathbf{L}$ is a set of literals closed under instantiation, and $Cond$ is a certain condition to be satisfied, e.g., the maximum length of clauses, the maximum depth of terms, etc. When $Cond$ is not specified, $\mathcal{P} = \langle \mathbf{L}, \emptyset \rangle$ is simply denoted as $\mathbf{L}$. A clause *C belongs to $\mathcal{P} = \langle \mathbf{L},\ Cond \rangle$* if every literal in C belongs to $\mathbf{L}$ and C satisfies $Cond$. For a set Σ of clauses, the set of logical consequence of Σ belonging to $\mathcal{P}$ is denoted as $Th_{\mathcal{P}}(\Sigma)$. Then, the *characteristic clauses* of Σ with respect to $\mathcal{P}$ are defined as:

$$Carc(\Sigma, \mathcal{P}) = \mu\, Th_{\mathcal{P}}(\Sigma)\,. \tag{3}$$

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 and $\mathcal{P}$ is a *stable* production field.[1] This means that proof finding is a special case of consequence finding.

The use of characteristic clauses enables us to characterize various reasoning problems of interest to AI, such as nonmonotonic reasoning, diagnosis, and knowledge compilation as well as abduction (Inoue, 1992, 2002). 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 clause. Formally, the *new characteristic clauses* of C with respect to Σ and $\mathcal{P}$ are:

$$\mathit{NewCarc}(\Sigma, C, \mathcal{P}) = \mu\,[\,Th_{\mathcal{P}}(\Sigma \wedge C) - Th(\Sigma)\,]. \tag{4}$$

It is shown in Inoue (1992, Proposition 2.7) that the definition (4) is equivalent to

$$\mathit{NewCarc}(\Sigma, C, \mathcal{P}) = \mathit{Carc}(\Sigma \wedge C, \mathcal{P}) - \mathit{Carc}(\Sigma, \mathcal{P}).$$

Example 2.1. The axioms Σ of an associative system with a left inverse and a left identity are given as (Lee, 1967):

$$\begin{array}{l}\neg p(x, y, u) \vee \neg p(y, z, v) \vee \neg p(x, v, w) \vee p(u, z, w),\\ \neg p(x, y, u) \vee \neg p(y, z, v) \vee \neg p(u, z, w) \vee p(x, v, w),\\ p(e, x, x),\\ p(i(x), x, e).\end{array}$$

By putting the production field as

$$\mathcal{P} = \langle\, \{p(\_, \_, \_)\}^{+},\ \text{length} \leq 1 \text{ and term-depth} \leq 1\,\rangle,$$

where $\{p(\_, \_, \_)\}^{+}$ is the set of all positive literals whose predicate symbol is p. If we choose C_1 as the first clause of Σ, then we obtain the new characteristic clauses $N = \mathit{NewCarc}(\Sigma - \{C_1\}, C_1, \mathcal{P})$ as

$$\begin{aligned}N = {} & p(x, i(x), e)) \wedge p(x, e, x) \wedge p(e, e, i(e)) \wedge \\ & p(i(x), x, i(e)) \wedge p(i(e), x, x) \wedge p(i(e), i(e), e).\end{aligned}$$

Note here that we do not have the axiom $i(e) = e$. The first and second clauses in N are the desired ones, which represent that the left inverse is also a right inverse and that the left identity is also a right identity.

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 where each of the added new formulas is a single clause (Inoue, 1992, Proposition 2.8):

$$\mathit{NewCarc}(\Sigma, F, \mathcal{P}) = \mu \left[\bigwedge_{i=1}^{m} \mathit{NewCarc}(\Sigma_i, C_i, \mathcal{P}) \right], \tag{5}$$

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.

$$\mathit{Carc}(\Sigma, \mathcal{P}) = \mathit{NewCarc}(\mathit{True}, \Sigma, \mathcal{P}). \tag{6}$$

The Eqs. (5) and (6) are independent of the order of the clauses in F and Σ. For example, when $\Sigma = (\neg p \vee q) \wedge p$,

$$\begin{aligned}\mathit{Carc}(\Sigma, \mathcal{L}) &= \mathit{NewCarc}(\mathit{True}, (\neg p \vee q) \wedge p, \mathcal{L})\\ &= \mu\,[\mathit{NewCarc}(\mathit{True}, \neg p \vee q, \mathcal{L}) \wedge \mathit{NewCarc}(\neg p \vee q, p, \mathcal{L})]\\ &= \mu\,[(\neg p \vee q) \wedge (p \wedge q)]\\ &= p \wedge q,\end{aligned}$$

where $\mathcal{L} = \{p, \neg p, q, \neg q\}$, and similarly,

$$\begin{aligned}\mathit{Carc}(\Sigma, \mathcal{L}) &= \mathit{NewCarc}(True, p \wedge (\neg p \vee q), \mathcal{L})\\ &= \mu\,[\mathit{NewCarc}(True, p, \mathcal{L}) \wedge \mathit{NewCarc}(p, \neg p \vee q, \mathcal{L})]\\ &= \mu\,[p \wedge q]\\ &= p \wedge q.\end{aligned}$$

Several procedures have been proposed to compute (new) characteristic clauses. For example, *SOL resolution* (Inoue, 1992) 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 (Iwanuma, Inoue, & Satoh, 2000). Instead of ME, *SFK resolution* (del Val, 1999) 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 (2000).

3. Inverting entailment for abduction and induction

This section reviews previous approaches to hypothesis finding through inverse entailment for abduction and induction. Recall that the logical setting of both inductive logic programming (ILP) and abductive logic programming (ALP) is given as follows.

Input: B: a background theory
E: (positive) examples/observations
Output: H: a hypothesis satisfying

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

Inverse entailment (IE) is defined as follows. Given that $B \wedge H \models E$, computing a hypothesis H can be done via the relation:

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

That is, the negation of H is entailed by $B \wedge \neg E$.
For IE, we should consider the following three problems.

1. *Computation*. How to compute $\neg H$ via $B \wedge \neg E \models \neg H$?
2. *Class*. Which classes of theories are allowed for B, E, and H?
3. *Completeness*. Is the calculus sound and complete for generating hypotheses?

These three problems are considered for both abduction and induction in the following subsections.

3.1. Inverse entailment for abduction

Computing hypotheses in abduction via IE is considered in Inoue (1992). Abduction is elegantly characterized by consequence finding as follows. We here denote the set of all literals in the representation language by $\mathcal{L}$, and a set Γ of candidate hypotheses is defined as a subset of $\mathcal{L}$. Any subset H of Γ is identified with the conjunction of all elements in H. Also, for any set T of formulas, $\overline{T}$ represents the *opposite* of T, which is the set of formulas obtained by negating every formula in T, i.e., $\overline{T} = \{\neg C \mid C \in T\}$.

Let $E_1, \ldots, E_n$ be a finite number of observations, and suppose that they are all literals. We want to explain the observations $E = E_1 \wedge \cdots \wedge E_n$ from an *abductive theory* (B, Γ), where B is a clausal theory representing a background theory and Γ is a set of ground literals representing an abductive bias. Then, $H = H_1 \wedge \cdots \wedge H_k$ is an *(abductive) explanation* of E from (B, Γ) if:

1. $B \wedge (H_1 \wedge \cdots \wedge H_k) \models E_1 \wedge \cdots \wedge E_n$,
2. $B \wedge (H_1 \wedge \cdots \wedge H_k)$ is consistent,
3. Each H_i is an element of Γ.

An explanation H of E is *minimal* if if no proper sub-conjunction H' of H satisfies $B \wedge H' \models E$. For minimal explanations, the following result holds.

Theorem 3.1. *Let (B, Γ) be an abductive theory. The set of minimal explanations of an observation E from (B, Γ) is*:

$$\overline{NewCarc(B, \neg E, \mathcal{P})}\,,$$

where the production field $\mathcal{P}$ is $\overline{\Gamma}$.

Proof: This theorem is essentially the same as Inoue (1992, Proposition 3.2), and we here give an outline of the proof. Suppose that $H = H_1 \wedge \cdots \wedge H_k$ is an abductive explanation of E from (B, Γ). Then, the above three conditions for E are equivalent to the following:

1′. $B \wedge (\neg E_1 \vee \cdots \vee \neg E_n) \models \neg H_1 \vee \cdots \vee \neg H_k$,
2′. $B \not\models \neg H_1 \vee \cdots \vee \neg H_k$,
3′. Each $\neg H_i$ is an element of $\overline{\Gamma}$.

By 1′, a clause derived from the clausal theory $B \wedge \neg E$ is the negation of an explanation of E from (B, Γ). By 2′, such a derived clause must not be a consequence of B before adding $\neg E$. By 3′, every literal appearing in such a clause must belong to $\overline{\Gamma}$. Moreover, H is a minimal explanation from (B, Γ) if and only if $\neg H$ is such a minimal consequence from $B \wedge \neg E$. Hence, the theorem holds. □

Hence, the problem of abduction is reduced to the consequence finding problem which seeks a clause C such that (i) C is a minimal consequence of $B \wedge \neg E$, but (ii) C is not a consequence of B alone, and (iii) C consists of literals only from the production field $\overline{\Gamma}$. Note here that *both $\neg E$ and $\neg H$ are clauses*. Hence, a resolution-based consequence finding procedure can be used to deduce $\neg H$ from $B \wedge \neg E$.

In the above setting, E is assumed to be a conjunction of literals. Extending the form of each example E_i to a clause, let $E = E_1 \wedge \cdots \wedge E_n$ be a CNF formula, where each E_i is a clause. Then, $\neg E$ is a DNF formula. By converting $\neg E$ from DNF into the CNF formula F, $NewCarc(B, F, \mathcal{P})$ can be computed by (5).

In Theorem 3.1, explanations obtained by a consequence finding procedure are not necessarily ground and can contain variables. In implementing resolution-based abductive procedures, however, each variable in the CNF formula E is replaced with a new constant in the complement $\neg E$ through Skolemization. To get a universally quantified explanation by negating each new characteristic clause containing Skolem constants, we need to apply the *reverse Skolemization* algorithm (Cox & Pietrzykowski, 1986). For example, if $\neg P(x, sk_y, u, sk_v)$ is a new characteristic clause where sk_y, sk_v are Skolem constants, we get the explanation $\forall y \forall v \exists x \exists u\, P(x, y, u, v)$ by reverse Skolemization.

To summarize, abduction via IE gives us the following answers to the three problems for IE.

IE for Abduction (Inoue, 1992)

1. *Computation*. The negation of a hypothesis, $\neg H$, is computed using a consequence finding procedure.

2. *Class*. In the basic setting, we consider

   B: a full clausal theory (containing non-Horn clauses),
   E: a conjunction of (existentially-quantified) literals,
   H: a conjunction of literals (belonging to the abductive bias).

3. *Completeness*. For the above class, the IE calculus is sound and complete for computing abductive explanations.

3.2. *Inverse entailment for induction*

Computing hypotheses in induction via IE was first considered by Muggleton (1995) in the Progol system. Muggleton considered a Horn program for a background theory B, a single Horn clause as an example E, and a single Horn clause as a hypothesis H. Even in this setting, however, *neither* $\neg E$ *nor* $\neg H$ *is a single clause*. Moreover, *both* $\neg E$ *and* $\neg H$ *contain existentially quantified variables*. This means that a simple application of a resolution-based procedure is not sufficient to compute inductive hypothesis. Then, Muggleton (1995) introduced the *bottom clause*:

$$\bot(B, E) = \{\neg L \mid L \text{ is a literal and } B \wedge \neg E \models L\}, \tag{7}$$

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

$$H \models \bot(B, E). \tag{8}$$

Any hypothesis H obtained in this way is correct, that is, it satisfies $B \wedge H \models E$. Yamamoto (2000) used two special consequence finding procedures to implement IE in this way. However, Yamamoto (1997) also showed that this method is incomplete for finding inductive hypotheses (see Example 4.3 in this paper). Sufficient conditions for the completeness to hold have been investigated in Yamamoto (1997) and Furukawa (1997). See the details of these previous works in Section 6.

Hence, induction via IE by Muggleton (1995) can be summarized as follows.

IE for Induction (Muggleton, 1995)

1. *Computation*. The negation of a hypothesis, $\neg H$, is computed using consequence finding procedures (Yamamoto, 2000).
2. *Class*. In the original setting, we consider

   B: a Horn program,
   E: a Horn clause,
   H: a Horn clause.

3. *Completeness*. For the above class, the IE calculus is sound but incomplete for computing inductive hypotheses (Yamamoto, 1997).

In the next section, we introduce a new approach to IE called *CF-Induction*. Instead of computing the bottom clause $\bot(B, E)$, CF-induction computes characteristic clauses of $B \wedge \neg E$, hence any resolution-based consequence finding procedure can be used. CF-induction includes all previous approaches to IE as special cases, in particular, includes abductive computation.

The specification of CF-induction is as follows. The details are given in Section 4.

CF-Induction

1. *Computation*. A consequence finding procedure is used.
2. *Class*. The most general class is considered:

   B: a full clausal theory,
   E: a full clausal theory,
   H: a full clausal theory.

3. *Completeness*. The IE calculus is sound and complete for computing inductive hypotheses.

4. Induction as consequence finding

In this section, we characterize explanatory induction by consequence finding.

4.1. CF-induction

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

$$B \wedge \neg E \models \neg H, \tag{9}$$
$$B \not\models \neg H. \tag{10}$$

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 negation of the bottom clause $\bot(B, E)$ in Muggleton (1995), we consider some clausal theory $CC(B, E)$ as a "bridge" formula U. Then, Eq. (9) can be written as

$$B \wedge \neg E \models CC(B, E), \tag{11}$$
$$CC(B, E) \models \neg H. \tag{12}$$

The latter (12) is also written as

$$H \models \neg CC(B, E). \tag{13}$$

Also, by (10) and (12), we have

$$B \not\models CC(B, E). \tag{14}$$

By (11), $CC(B, E)$ is obtained by computing the characteristic clauses of $B \wedge \neg E$ because any other consequence of $B \wedge \neg E$ belonging to $\mathcal{P}$ can be obtained by constructing a clause that is subsumed by a characteristic clause. Hence,

$$\mathit{Carc}(B \wedge \neg E, \mathcal{P}) \models CC(B, E), \tag{15}$$

where the production field $\mathcal{P} = \langle \mathbf{L}, \mathit{Cond} \rangle$ is defined as a pair of a set $\mathbf{L}$ of literals reflecting an *inductive bias* in the complement form and a certain condition Cond. When no inductive bias is considered, $\mathcal{P}$ is just set to $\mathcal{L}$, which is the set of all literals in the first-order language. The other requirement for $CC(B, E)$ is Eq. (14), which is satisfied if at least one of the 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 $\mathit{NewCarc}(B, \neg E, \mathcal{P})$ in $CC(B, E)$.

In constructing a hypothesis H from the clausal theory $CC(B, E)$, notice that $\neg CC(B, E)$ is entailed by H in (13). Since $\neg CC(B, E)$ is DNF, we convert it into the CNF formula F, i.e.,

$$F \equiv \neg CC(B, E). \tag{16}$$

Then, H is constructed as a clausal theory which entails F, i.e.,

$$H \models F. \tag{17}$$

In ILP, there are several methods to compute a new clausal theory H which entails a given clausal theory F. A procedure to construct such a more general clausal theory is called a *generalizer* (Yamamoto & Fronhöfer, 2000) (see Section 4.2). Note that applying an 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 $\mathit{NewCarc}(B, \neg E, \mathcal{P})$.

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

Definition 4.1. 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 $\mathit{Carc}(B \wedge \neg E, \mathcal{P})$;

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

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.

Several remarks are necessary for the definition of CF-induction.

1. At Step 1, the number of characteristic clauses in $Carc(B \wedge \neg E, \mathcal{P})$ may be large or infinite in general. Hence, this step should be interleaved on demand with construction of each C_i at Step 2 in practice.
2. At Step 2, a selected clause C_i in $CC(B, E)$ can contain variables. In this case, each variable x in C_i is replaced with a Skolem constant sk_x in the complement $\neg C_i$ at Step 3, in which x is interpreted as existentially quantified. Sometimes we need multiple Skolem constants $sk_x^1, sk_x^2, \cdots$ for each variable x of $\neg C_i$, depending on how many times C_i is used in deriving $\neg H$ from $B \wedge \neg E$. See 7 in the appendix for further details and an example.
3. At Step 3, a DNF formula $\neg CC(B, E)$ is converted to CNF. The complexity of this computation is high, that is, in the class #P (Yamamoto & Fronhöfer, 2000). Of course, we do not need this conversion if we allow a DNF hypothesis as H.

4.2. Generalizers

At Step 4 of CF-induction, we need a *generalizer*. The task of a generalizer is, given a CNF formula F, to find a CNF formula H such that

$$H \models F.$$

There are several methods to realize a generalizer. For example, the following techniques are well-known, and can be jointly used as a generalizer.

- *Reverse Skolemization* (Cox & Pietrzykowski, 1986): Skolem constants/functions are converted to existentially quantified variables.
- *Anti-instantiation*: ground terms are replaced with variables.
- *Anti-subsumption (dropping)*: some literals are dropped from a clause.
- *Strengthening (anti-weakening)*: some clauses are added. Yamamoto (2001) argued that the application of anti-weakening might cause difficulties because any clausal theory F' that is a superset of F is derived as a correct hypothesis H with this operation. Hence, anti-weakening should be used in a restricted way so that H does not contain any clause which is not used to explain E.
  On the other hand, anti-weakening is necessary to assure the completeness of CF-induction (Theorem 4.4 in Section 4.4). For instance, any hypothesis which contain redundant clauses can be obtained only by anti-weakening. This fact indicates that a real implementation is incomplete if it avoids producing some redundant hypotheses.
- *Inverse resolution* (Muggleton & Buntine, 1988): the inverse of the resolution principle is applied. In some cases, this is reduced to the *folding* operation in logic programming (Pettorossi & Proietti, 1994). This operation is useful for introducing new predicates/literals not appearing in F.
- *Least generalization* (Plotkin, 1971): a least general generalization is constructed from multiple clauses.

Note that if the "entailment" relation $\models$ is replaced with the weaker "subsumption" relation in $H \models F$, the completeness in Theorem 4.1 (shown in Section 4.4) does not precisely hold. When a hypothesis H such that H subsumes F is found, $B \wedge H \models E$ does not necessarily hold, but it holds that *H subsumes E relative to B* in the sense of Plotkin (1971). See Yamamoto (1997) for details.

4.3. Examples

Example 4.1. For the introductory example shown in Section 1, $B_1 = human(s)$ and $E_1 = mortal(s)$. Then,

$$Carc(B_1 \wedge \neg E_1, \mathcal{L}) = human(s) \wedge \neg mortal(s),$$

where $\neg mortal(s)$ is the clause in $NewCarc(B_1, \neg E_1, \mathcal{L})$. In this case, $CC(B_1, E_1)$ is set to $Carc(B_1 \wedge \neg E_1, \mathcal{L})$. Then,

$$F_1 \equiv \neg CC(B_1, E_1) = \neg human(s) \vee mortal(s).$$

By applying anti-instantiation to F_1 with s/x, we get

$$H_1 = (human(x) \supset mortal(x)).$$

Example 4.2. The following theory is a variant of an example in Buntine, 1988, and is often used to illustrate how the bottom clause is used in inverse entailment (Yamamoto, 1997, 2000, 2001). Consider

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

Then, the complement of E_2 is

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

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, $F_2 = \neg CC(B_2, E_2)$ is equivalent to $\bot(B_2, E_2)$. By applying reverse Skolemization (or anti-instantiation) to F_2, we get the hypothesis:

$$H_2 = (fluffy(x) \wedge cat(x) \wedge pet(x) \supset cuddly\_pet(x) \vee small(x)).$$

While in the above cited references the subclause of H_2:

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

is often adopted as a definite clause, H_2 is the most-specific hypothesis in the sense of Muggleton (1995).

In Yamamoto (2001), the following hypothesis:

$$H_2' = (pet(x) \supset dog(x)) \wedge (dog(x) \supset small(x))$$

is shown as another correct hypothesis if dog is a predicate symbol in the language. This is obtained if we take $CC'(B_2, E_2) = pet(sk_x) \wedge \neg small(sk_x)$. That is, $F_2' = \neg CC'(B_2, E_2) = (pet(sk_x) \supset small(sk_x))$. Then, H_2' is constructed by applying anti-instantiation and inverse resolution (or folding).

Example 4.3. This example (Yamamoto, 1997) illustrates the incompleteness of inverse entailment based on the bottom clause in Muggleton (1995). Consider the background theory and the example:

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

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 second 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 a hypothesis. There are many ways to compute H_3 from F_3. For example, by computing the least generalization of the two clauses in F_3, we obtain the clause:

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

Dropping two ground literals from the above, we get H_3.

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. In fact, $H_3 \not\models \bot(B_3, E_3)$.

4.4. Completeness

We now present the correctness result for clausal theories derived using CF-induction. The result implies not only the completeness but also the soundness of CF-induction.

Theorem 4.1. *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 1, both B and E may contain non-Horn clauses and integrity constraints. Also, the derived hypothesis H may be non-Horn. This result answers the *open question* posed by Muggleton (1998), as to *whether a generalization of inverse entailment would be complete for arbitrary clausal background theories*.

As a special case of CF-induction, we can obtain an IE procedure based on the bottom clause $\bot(B, E)$. Notice here that for the production field with no restriction, i.e., $\mathcal{P} = \mathcal{L}$, it holds that

$$\overline{\bot(B, E)} \subseteq Carc(B \wedge \neg E, \mathcal{L}).$$

Instead of $\mathcal{L}$, consider now the production field

$$\mathcal{P}^{\leq 1} = \langle \mathcal{L}, \text{ length} \leq 1 \rangle.$$

Whenever the background theory B is consistent and $B \not\models E$, the characteristic clauses of $B \wedge \neg E$ with respect to $\mathcal{P}^{\leq 1}$ are equivalent to the opposite of the bottom clause, i.e.,

$$Carc(B \wedge \neg E, \mathcal{P}^{\leq 1}) = \overline{\bot(B, E)}.$$

Hence, the restricted version of CF-induction with the production field $\mathcal{P}^{\leq 1}$ is essentially the same as the procedure by Muggleton (1995). The soundness of CF-induction in this case is guaranteed for not only Horn programs but also arbitrary clausal theories B, although the completeness does not hold even for Horn programs as seen in Example 4.3.

Corollary 4.1. *Let B, E, and H be clausal theories. Suppose that the production field is set as $\mathcal{P}^{\leq 1}$. If H is derived by a CF-induction from B and E, then $B \wedge H \models E$ and $B \wedge H$ is consistent.*

4.5. *Negative examples*

So far, we have not considered negative examples in CF-induction. To avoid over-generalization, it is common for an ILP system to use negative examples as well as positive ones. For CF-induction, negative examples can also be incorporated as follows.

Suppose that B, E^+, and E^- are a background theory, positive examples, and a negative example, respectively. Here, we assume that there exists only one negative example, but extending the case to multiple negative examples is possible. The task of explanatory induction in this case is to compute a hypothesis H satisfying both

$$B \wedge H \models E^+$$

and

$$B \wedge H \not\models E^-.$$

Here, the negative example can be used to reduce the number of possible choices for $CC(B, E^+)$, which contributes toward increasing the efficiency of CF-induction as follows. Instead of (10), we now have

$$B \wedge \neg E^- \not\models \neg H,$$

and the relation (14) is replaced with

$$B \wedge \neg E^- \not\models CC(B, E^+).$$

Hence, it holds that

$$Carc(B \wedge \neg E^-, \mathcal{P}) \not\models CC(B, E^+).$$

Then, at Step 2(b) of CF-induction, an instance of a clause from $Carc(B \wedge \neg E^+, \mathcal{P}) - Carc(B \wedge \neg E^-, \mathcal{P})$ must be selected in $CC(B, E^+)$. Moreover, at Step 4, H must be constructed so that $B \wedge H \not\models E^-$.

4.6. *Implementation*

An incomplete version of CF-induction has been implemented in Java. The incompleteness of such a procedure is due to the incompleteness of generalizers. In particular, anti-weakening (clause addition) cannot be realized completely, but this is not practically bad because increasing clauses in H results in a redundant hypothesis in general (see Section 4.2). As a consequence finding procedure, we have realized a tableaux version of SOL resolution (Iwanuma, Inoue, & Satoh, 2000). After computing $Carc(B \wedge \neg E, \mathcal{P})$, the selection of $CC(B, E)$ is guided by a menu, in which a user choose clauses from $Carc(B \wedge \neg E, \mathcal{P})$.

At this time, the lack of a clause from $NewCarc(B, \neg E, \mathcal{P})$ is automatically detected. Moreover, if a clause with variables is selected for $CC(B, E)$, it is instantiated with terms constructed from constants and functions, in which the number of created terms is specified by a user in advance. As generalizers, we realized anti-instantiation, least generalization, and anti-subsumption. See Otsuji (2002) for details of a Java implementation of CF-induction.

Runtimes for examples shown in Section 4.3 are: 288 msec (Example 4.2) and 150 msec (Example 4.3) in a Pentium III machine. The reason why it takes more time for Example 4.2 is that the negation of E_2 is not a single clause but a conjunction of literals, so multiple SOL deductions have to be computed iteratively via (5).

5. Abduction vs. induction

CF-induction is realized by abductive computation. In fact, computing $Carc(B \wedge \neg E, \mathcal{P})$ at Step 1 can be implemented by calling *NewCarc* operations incrementally in (5) and (6), each of which can be regarded as computing abduction by Theorem 3.1.

Conversely, computing abduction is regarded as a special case of CF-induction.

Theorem 5.1. *Let (B, Γ) be an abductive theory. A conjunction H of literals is a minimal explanation of an observation E from (B, Γ) if and only if H is derived by a CF-induction from B and E in which the size of $CC(B, E)$ at Step 2 is 1 ($m = 1$) and reverse Skolemization is used as the generalizer at Step 4.*

Proof: When the size of $CC(B, E)$ is 1, $CC(B, E)$ consists of a single clause C taken from $NewCarc(B, \neg E, \mathcal{P})$ at Step 2. The negation of C is then a minimal explanation of E by Theorem 3.1. □

The set of all minimal explanations is characterized by Theorem 3.1, and can also be obtained by slightly modifying CF-induction. Namely, every clause of $CC(B, E)$ is taken from $NewCarc(B, \neg E, \mathcal{P})$ at Step 2, and we do not have to convert $\neg CC(B, E)$ into CNF at Step 3, and reverse Skolemization is used as the generalizer at Step 4. By Theorem 5.1, each single conjunction $\neg C_i$ obtained in this way is a minimal explanation of E. Then, the disjunction $\neg CC(B, E)$ of every $\neg C_i$ is also an explanation. Such DNF explanations are used in AI applications such as computing *circumscription* (Helft, Inoue, & Poole, 1991), *diagnosis* (Konolige, 1992), and knowledge base updates and knowledge assimilation (Inoue & Sakama, 2002).

Thus, abduction and induction are very similar if we allow arbitrary forms of clausal theories as hypotheses. There are three main differences between them.

1. By convention, the form of hypotheses in induction is CNF, while it is usually DNF (or a set of conjunctions) in abduction.
2. In induction, at least one of the clauses in $CC(B, E)$ is taken from $NewCarc(B, \neg E, \mathcal{P})$. On the other hand, all clauses in $CC(B, E)$ must be in $NewCarc(B, \neg E, \mathcal{P})$ in abduction (by Theorem 3.1).

3. Reverse Skolemization is solely used as a generalizer in abduction, while other generalizers can be used in induction. In particular, when anti-weakening is allowed as a generalizer, an obtained hypothesis is not necessarily minimal in induction, while minimal explanations are usually preferred in abduction.

No other difference exists between abduction and induction as long as their implementation is concerned in the context of consequence finding. The next example illustrates the similarity between induction and abduction.

Example 5.1. (Yamamoto & Fronhöfer, 2000). Let

$$B_4 = (dog(x) \wedge small(x) \supset pet(x)),$$
$$E_4 = pet(c).$$

be the background theory and the example. Then,

$$NewCarc(B_4, \neg E_4, \mathcal{L}) = \neg pet(c) \wedge (\neg dog(c) \vee \neg small(c)).$$

Now, put $CC(B_4, E_4) = NewCarc(B_4, \neg E_4, \mathcal{L})$. Then,

$$\neg CC(B_4, E_4) = pet(c) \vee (dog(c) \wedge small(c)),$$

which is exactly the same as the minimal abductive explanations. Converting $\neg CC(B_4, E_4)$ into CNF, we have

$$F_4 = (dog(c) \vee pet(c)) \wedge (small(c) \vee pet(c)).$$

By applying anti-instantiation, we get the clausal theory:

$$H_4 = (dog(x) \vee pet(x)) \wedge (small(x) \vee pet(x)).$$

On the other hand, the next example shows the main difference between abduction and induction, which is the second one in the above differences: *all clauses in $CC(B, E)$ are taken from NewCarc$(B, \neg E, \mathcal{P})$ in abduction, while it is not the case in induction.* Namely, induction often utilizes consequences of B before adding $\neg E$ in the construction of $CC(B, E)$. This operation is essential to associate observations E with the background theory B in induction. Abduction, on the other hand, does not need such consequences of B because they are redundant in virtue of the minimality of explanations. This difference also reflects Peirce's theory of induction and abduction (Peirce, 1932): induction infers a rule (i.e., hypothesis) $A \supset C$ from a case (i.e., background theory) A and a result (i.e., example/observation) C, while abduction infers a case A from a rule $A \supset C$ and a result C.

Example 5.2. (Muggleton, 1995). Let us consider the background theory and the example:

$$B_5 = \mathit{white}(\mathit{swan}1), \quad E_5 = \neg \mathit{black}(\mathit{swan}1).$$

Then, $\mathit{NewCarc}(B_5, \neg E_5, \mathcal{L}) = \neg E_5 = \mathit{black}(\mathit{swan}1)$. Hence, $\neg \mathit{black}(\mathit{swan}1)$ is the unique minimal abductive explanation of E_5. In induction, on the other hand, let

$$CC(B_5, E_5) = \mathit{white}(\mathit{swan}1) \wedge \mathit{black}(\mathit{swan}1),$$

in which the first conjunct is the clause of B_5. By anti-instantiating $F_5 = \neg CC(B_5, E_5)$, we can learn the integrity constraint:

$$H_5 = \neg \mathit{white}(x) \vee \neg \mathit{black}(x).$$

6. Related work

6.1. Previous work on inverse entailment

CF-induction is obviously influenced by previous work on inverse entailment (IE). As shown in Section 3.2, the original IE (Muggleton, 1995) allows Horn clauses for B and a single Horn clause for each of H and E. Even in this setting, however, IE based on $\bot(B, E)$ is incomplete for finding H such that $B \wedge H \models E$ (Yamamoto, 1997). Furukawa et al. (1997) consider a sufficient condition for IE to be complete, which restricts the class of logic programs to a proper subset of the Horn programs. Muggleton (1998) considers an enlarged bottom set to make IE complete in the class of Horn programs, but the revised method is unsound. Furukawa (1998) also proposes a complete algorithm, but it is relatively complex. Yamamoto (2000) 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 (2000) suggest the use of PTTP (Stickel, 1988) 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 works, CF-induction proposed in this paper is simple, yet sound and complete for finding hypotheses from not only Horn programs but also full clausal theories. Instead of the bottom clause, CF-induction uses the characteristic clauses, which strictly include the unit clauses in $\overline{\bot(B, E)}$.

6.2. Yamamoto and Fronhöfer

Yamamoto and Fronhöfer (2000) first extend IE to allow for full clausal theories for B and E, and introduce the notion of a *residue hypothesis* for a set of ground instances of $B \wedge \neg E$. A residue hypothesis is computed from the ground instances T of $B \wedge \neg E$ by:

1. Select a finite set (i.e., conjunction) S of ground clauses from T;

2. Convert $\neg S$ into the CNF formula R;
3. Remove all tautologies from R.

An inductive hypothesis H is then computed by applying a generalizer to a residue hypothesis.

A residue hypothesis can be computed using Bibel's *Connection method* (Bibel, 1993), and the constructed CNF formula R corresponds to the enumeration of all paths in the matrix of clauses S. 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 Yamamoto and Fronhöfer, a merit of CF-induction lies in the existence of a *production field* $\mathcal{P}$, which can be used to guide and restrict derivations of clauses by reflecting an *inductive bias*. Usually, we are given some inductive bias for inductive problems. For example, the production field $\mathcal{P}^{\leq 1}$ in Section 4.4 guides derivations of a consequence finding procedure to produce unit clauses only. Also, Progol (Muggleton, 1995) uses mode declarations to constrain search for H which subsumes $\bot(B, E)$. It is unclear how to incorporate inductive biases in the procedure of Yamamoto and Fronhöfer (2000).

Note that the consistency of a residue hypothesis is not always guaranteed. Although not mentioned in Yamamoto and Fronhöfer (2000), the consistency is assured if instances of $\neg E$ are selected within S from the ground instances T of $B \wedge \neg E$, which is similar to our incorporation of $\mathit{NewCarc}(B, \neg E, \mathcal{P})$ into $CC(B, E)$.

Finally, residue hypotheses are relatively longer and more complex than hypotheses constructed by CF-induction, and require more efforts for a generalizer to construct a final inductive hypothesis H.

Example 6.1. Suppose that we are given the background theory and an example as:

$$B_6 = (a \vee b) \wedge (a \supset c) \wedge (b \supset c) \wedge (d \supset g),$$
$$E_6 = g.$$

Then,

$$\mathit{NewCarc}(B_6, \neg E_6, \mathcal{L}) = \neg d \wedge \neg g,$$
$$\mathit{Carc}(B_6 \wedge \neg E_6, \mathcal{L}) = \mathit{NewCarc}(B_6, \neg E_6, \mathcal{L}) \wedge (a \vee b) \wedge c.$$

If we put $CC(B_6, E_6) = \mathit{Carc}(B_6 \wedge \neg E_6, \mathcal{L})$, we get a hypothesis:

$$H_6 = (a \wedge c \supset d \vee g) \wedge (b \wedge c \supset d \vee g).$$

Alternatively, suppose we take another bridge as

$$CC'(B_6, E_6) = c \wedge \neg d.$$

Then,

$$F'_6 = \neg CC'(B_6, E_6) = (c \supset d).$$

This formula F'_6 is taken as an inductive hypothesis H'_6 without applying any generalizer.

On the other hand, it is not easy to obtain $H'_6 = (c \supset d)$ using Yamamoto and Fronhöfer (2000) method. Firstly, all clauses of $B_6 \wedge \neg E_6$ are necessary to construct H'_6. Then, the residue hypothesis for $B_6 \wedge \neg E_6$ is

$$\begin{aligned} &(a \wedge c \supset b \vee d \vee g) \wedge (a \wedge c \supset d \vee g) \\ \wedge\, &(b \wedge c \supset a \vee d \vee g) \wedge (b \wedge c \supset d \vee g) \end{aligned}$$

In the above formula, the first and the third clauses are subsumed by the second and fourth clauses, respectively, and are redundant. After removing them, take the least generalization of $(a \wedge c \supset d \vee g)$ and $(b \wedge c \supset d \vee g)$, which is $(c \supset d \vee g)$. Finally, g is dropped from this clause to construct $(c \supset d)$. Hence, subsumption tests must be used for simplifying residue hypotheses. In CF-induction, the notion of subsumption tests is already implicit in the notion of characteristic clauses.

7. Conclusion and future work

In this paper, we have defined a general resolution-based method to construct inductive hypotheses from full clausal theories. We put emphasis on finding a sound and complete method for inverse entailment in full clausal theories, which was a long-standing open problem in ILP. To this problem, we have suggested a simple yet powerful solution: CF-induction. Salient features of CF-induction are summarized as follows.

- CF-induction 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 based on the resolution principle.
- CF-induction can be implemented with existing systematic consequence finding procedures such as SOL resolution (Inoue, 1992) and SFK resolution (del Val, 1999).
- CF-induction includes, as special cases, computing abductive explanations and the bottom clause. Also, a restriction on the hypothesis vocabulary can easily be realized by specifying a production field.

We also clarified the similarity and difference between abduction and induction in the context of consequence finding.

CF-induction by Definition 4.1 computes the characteristic clauses $Carc(B \wedge \neg E, \mathcal{P})$, selects $CC(B, E)$ to be a set of instances of $Carc(B \wedge \neg E, \mathcal{P})$, and then applies a generalizer to the complement of $CC(B, E)$ to obtain H. There are two choice points in this method: the choice of $CC(B, E)$, and the choice of the generalizer. Currently, these two choice points cannot be combined into one. However, an open question is whether or when it is possible to construct a *single* bridge formula U such that any H can be derived in the complement form from U using a generalizer. A simple method for such an ultimate bridge would be to include all clauses from $Carc(B \wedge \neg E, \mathcal{P})$ into $CC(B, E)$. However, this only works when (the instances of) $Carc(B \wedge \neg E, \mathcal{P})$ is finite, and even in the finite case, a generalizer should work very hard! In other words, a heavy work load of a generalizer is reduced if an appropriate choice is made in the selection of $CC(B, E)$.

A prototype system of CF-induction has been implemented in Java, but efficient implementation of CF-induction is an important future work. In particular, an intelligent selection of $CC(B, E)$ from $Carc(B \wedge \neg E, \mathcal{P})$ needs to be addressed. To remove the manual intervention in this step from the system, one can perform a search in the space of subsets of $Carc(B \wedge \neg E, \mathcal{P})$ or the space of possible instantiations of the clauses. In such a case, we need heuristics for guiding searches, e.g., compression and the description length. Moreover, testing the (extended) system with intelligent search on large and practical problems of learning from positive examples is necessary in the future, as for those examples used in Muggleton (2001). Learning from both positive and negative examples should also be automated with search in the framework of CF-induction. These extensions are necessary for the algorithm of CF-induction to construct a practical ILP system. However, we should again put emphasis on the completeness of CF-induction in extended systems. Making an ILP system complete is worthwhile to discover an interesting hypothesis even if it takes much time.

Finally, there exist formalizations of induction other than explanatory induction in the literature on ILP, such as *learning from interpretations (or satisfiability)* (De Raedt, 1997), and *descriptive induction* (Helft, 1989; Lachiche, 2000). De Raedt (1997) proposes a translation of learning from interpretations into learning from entailment, but the method requires negative examples. Lachiche (2000) discusses various forms of descriptive induction, which can also be characterized by deduction from *completed theories*. The precise relationships between these different formalisms and consequence finding need to be addressed in the future.

Appendix 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 7 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 7 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. The following proofs can be applied to the case with an inductive bias. In this case, the literals specified in the production field $\mathcal{P}$ are set to the opposite of $\mathcal{L}_H$. When $\mathcal{L}_H$ is the set of all clauses, $\mathcal{P}$ is given as $\mathcal{L}$. Note that a length restriction in a production field cannot be used to assure the completeness (e.g., $\mathcal{P}^{\leq 1}$ in Section 4.4). We also assume the existence of a sound and complete generalizer at Step 4 of a CF-induction.

A.1 Soundness of CF-induction

Let H be a hypothesis obtained by a CF-induction from B and E. By the definition of a CF-induction, there is a CNF 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 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 an instance of a clause from $NewCarc(B, \neg E, \mathcal{P})$. 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,

$$B \wedge \neg E \models C_1 \wedge \cdots \wedge C_m \quad \text{and} \quad B \not\models C_1 \wedge \cdots \wedge C_m.$$

Now, let

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

Then,

$$B \wedge \neg E \models \neg F \quad \text{and} \quad B \not\models \neg F,$$

which are equivalent to

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

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 $B \wedge H \models E$ and $B \wedge H$ is consistent. Then, $B \wedge \neg E \models \neg H$ and $B \not\models \neg H$. Since H belongs to $\mathcal{L}_H$ that is the opposite of the literals in $\mathcal{P}$, $\neg H$ belongs to $\mathcal{P}$. By the definition of the characteristic clauses, any consequence of $B \wedge \neg E$ belonging to $\mathcal{P}$ is subsumed by a clause in $Carc(B \wedge \neg E, \mathcal{P})$. 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: one uses Herbrand's theorem, and the other uses the compactness theorem.

Theorem A.1 (*Herbrand's Theorem*). *A set of clauses Σ is unsatisfiable if and only if a finite set of ground instances of clauses of Σ is unsatisfiable.*

Theorem A.2 (*Compactness*). *Let Σ be a set of clauses. If all finite subsets of Σ is satisfiable, then so is Σ. Equivalently, if Σ is unsatisfiable then a finite subset of Σ is unsatisfiable.*

A.2.1 **[A]** ***Using 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*$(B \wedge \neg E, \mathcal{P})$,
(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 Eq. (15) in Section 4). Finally, at Steps 3 and 4, H can be obtained by applying a generalizer (including anti-instantiation) to the CNF representation of $\neg CC(B, E)$.

A.2.2 **[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 $B \wedge \neg E$. Then, at Steps 3 and 4, H can be obtained by applying a generalizer to the CNF representation of $\neg CC(B, E)$.

Example A.1. We now verify the completeness proof [B] by applying it to a variant of Example 4.3 from Yamamoto and Fronhöfer (2000, Example 2), while the proof [A] can easily be checked in Example 4.3. Let us consider the background theory and the example:

$$B_7 = even(0) \wedge (\neg odd(x) \vee even(s(x))),$$
$$E_7 = odd(s^5(0)),$$

where $s^1(0) = s(0)$ and $s^n(0) = s(s^{n-1}(0))$ for $n > 1$. This time, we choose a non-ground characteristic clause as

$$CC(B_7, E_7) = even(0) \wedge (\neg odd(x) \vee even(s(x))) \wedge \neg odd(s^5(0)).$$

By making two copies of $\exists x(odd(x) \wedge \neg even(s(x)))$, the complement of $CC(B_7, E_7)$ becomes

$$\begin{aligned}
&\left(\neg even(0) \vee odd\left(sk_x^1\right) \vee odd\left(sk_x^2\right) \vee odd(s^5(0))\right) \\
&\wedge \left(\neg even(0) \vee odd\left(sk_x^1\right) \vee \neg even\left(s\left(sk_x^2\right)\right) \vee odd(s^5(0))\right) \\
&\wedge \left(\neg even(0) \vee \neg even\left(s\left(sk_x^1\right)\right) \vee odd\left(sk_x^2\right) \vee odd(s^5(0))\right) \\
&\wedge \left(\neg even(0) \vee \neg even\left(s\left(sk_x^1\right)\right) \vee \neg even\left(s\left(sk_x^2\right)\right) \vee odd(s^5(0))\right),
\end{aligned}$$

which represents

$$\begin{aligned}
\exists y \exists z\ [\ &(\neg even(0) \vee odd(y) \vee odd(z) \vee odd(s^5(0))) \wedge \\
&(\neg even(0) \vee odd(y) \vee \neg even(s(z)) \vee odd(s^5(0))) \wedge \\
&(\neg even(0) \vee \neg even(s(y)) \vee odd(z) \vee odd(s^5(0))) \wedge \\
&(\neg even(0) \vee \neg even(s(y)) \vee \neg even(s(z)) \vee odd(s^5(0)))\ \ ].
\end{aligned}$$

The hypothesis

$$H_7 = \neg even(x) \vee odd(s(x))$$

entails $\neg CC(B_7, E_7)$. To see this, take the substitution $\{y/s(0), z/s^3(0)\}$ in the above formula. Then, H_7 subsumes each clause in

$$\begin{aligned}
&(\neg even(0) \vee odd(s(0))) \\
&\wedge (\neg even(s^2(0)) \vee odd(s^3(0))) \\
&\wedge (\neg even(s^4(0)) \vee odd(s^5(0))),
\end{aligned}$$

and thus entails $\neg CC(B_7, E_7)$.

Acknowledgments

In 1995, Koichi Furukawa first suggested me to use a consequence finding procedure for computing inverse entailment. In 1997, Akihiro Yamamoto succeeded in realizing such an attempt for Horn programs. I am indebted to them for their previous works and stimulative discussions on improvement of IE calculi. I would like to thank reviewers for this paper and my ILP'01 paper for their helpful comments. I am grateful to Kanako Otsuji, Yoshitaka Yamamoto, and Hidetomo Nabeshima for their efforts at implementing CF-induction. Thanks also for useful discussions and comments on the topics in this paper with Laurence Cholvy, Philip Reiser, Haruka Saito, and Hiromasa Haneda.

Note

1. A production field $\mathcal{P}$ is *stable* if, for any two clauses C and D such that C subsumes D, D belongs to $\mathcal{P}$ only if C belongs to $\mathcal{P}$.

References

Bibel, W. (1998). *Deduction: Automated logic*. Academic Press.

Buntine, W. (1988). Generalized subsumption and its applications to induction and redundancy. *Artificial Intelligence*, *36*, 149–176.

Cox, P. T., & Pietrzykowski, T. (1986). Causes for events: their computation and applications. In *Proceedings of the Eighth International Conference on Automated Deduction* (pp. 608–621). LNCS 230, Springer.

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

del Val, A. (1999). A new method for consequence finding and compilation in restricted languages. In *Proceedings of AAAI-99* (pp. 259–264). AAAI Press.

Dimopoulos, Y., & Kakas, A. (1996). Abduction and inductive learning. In L. De Raedt (Ed.), *Advances in inductive logic programming*. IOS Press.

Flach, P. A., & Kakas, A. C. (Eds.). (2000). *Abduction and induction: Essays on their relation and integration*. Kluwer.

Furukawa, K., Murakami, T., Ueno, K., Ozaki, T., & K. Shimazu. (1997). On a sufficient condition for the existence of most specific hypothesis in Progol. In N. Lavrač, & S. Džeroski (Eds.), *Proceedings of the Seventh International Workshop on Inductive Logic Programming* (pp. 157–164). LNAI 1297, Springer.

Furukawa, K. (1998). On the completion of the most specific hypothesis computation in inverse entailment for mutual recursion. In *Proceedings of Discovery Science '98* (pp. 315–325). LNAI 1532, Springer.

Grégoire, É., & Saïs, L. (1996). Inductive reasoning is sometimes deductive. In *Proceedings of ECAI-96 Workshop on Abductive and Inductive Reasoning*.

Helft, N. (1989). Induction as nonmonotonic inference. In R.J. Brachman, H.J. Levesque, & R. Reiter (Eds.), *Proceedings of the First International Conference on Principles of Knowledge Representation and Reasoning* (pp. 149–156). Morgan Kaufmann.

Helft, N., Inoue, K., & Poole, D. (1991). Query answering in circumscription. In *Proceedings of IJCAI-91* (pp. 426–431). Morgan Kaufmann.

Inoue, K. (1992). Linear resolution for consequence finding. *Artificial Intelligence*, *56*, 301–353.

Inoue, K. (2001). Induction, abduction, and consequence-finding. In C. Rouveirol, & M. Sebag (Eds.), *Proceedings of the Eleventh International Conference on Inductive Logic Programming* (pp. 65–79). LNAI 2157, Springer.

Inoue, K. (2002). Automated abduction. In A. Kakas, & F. Sadri (Eds.), *Computational Logic: From Logic Programming into the Future—In Honour of Bob Kowalski* (Part II). LNAI 2408, Springer.

Inoue, K., & C. Sakama. (2002). Disjunctive explanations. In P. Stuckey (Ed.), *Proceedings of the Seventeenth International Conference on Logic Programming* (pp. 317–332), LNCS 2401, Springer.

Iwanuma, K., Inoue, K., & Satoh, K. (2000). Completeness of pruning methods for consequence finding procedure SOL. In P. Baumgartner, & H. Zhang (Eds.), *Proceedings of the Third International Workshop on First-Order Theorem Proving* (pp. 89–100).

Konolige, K. (1992). Abduction versus closure in causal theories. *Artificial Intelligence*, *53*, 255–272.

Lachiche, N. (2000). Abduction and induction from a non-monotonic reasoning perspective. In P.A. Flach, & A.C. Kakas (Eds.), *Abduction and induction: Essays on their relation and integration. Kluwer*.

Lee, C. T. (1967). 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.

Marquis, P. (2000). Consequence finding algorithms. In D. M. Gabbay & P. Smets (Eds.), *Handbook for defeasible reasoning and uncertain management systems* (vol. 5). Kluwer Academic.

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

Muggleton, S. (1998). Completing inverse entailment. In D. Page (Ed.), *Proceedings of the Eighth International Conference on Inductive Logic Programming* (pp. 245–249). LNAI 1446, Springer.

Muggleton, S. (2001). Learning from positive data. *Machine Learning*, to appear.

Muggleton, S., & Bryant, C. (2000). Theory completion and inverse entailment. In J. Cussens, & A. Frisch (Eds.), *Proceedings of the Tenth International Conference on Inductive Logic Programming* (pp. 130–146). LNAI 1866, Springer.

Muggleton, S., & Buntine, W. (1988). Machine invention of first-order predicates by inverting resolution. In *Proceedings of the Fifth International Conference on Machine Learning* (pp. 339–352). Morgan Kaufmann.

Nienhuys-Cheng, S. H., & de Wolf, R. (1997). *Foundations of inductive logic programming*. LNAI 1228, Springer.

Otsuji, K. (2002). Research on implementation of CF-induction using SOL resolution. Graduation Thesis, Department of Electrical and Electronics Engineering, Kobe University.

Peirce, C. S. (1932). Elements of logic. In C. Hartshorne, & P. Weiss (Eds.), *Collected papers of Charles Sanders Peirce* (vol. II). Cambridge, MA: Harvard University Press.

Pettorossi, A., & 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, & D. Michie (Eds.), *Machine intelligence* (vol. 6). Edinburgh University Press.

Stickel, M. E. (1988). A Prolog technology theorem prover: implementation by an extended Prolog compiler. *Journal of Automated Reasoning*, *4*, 353–380.

Yamamoto, A. (1997). Which hypotheses can be found with inverse entailment? In N. Lavrač, & S. Džeroski (Eds.), *Proceedings of the Seventh International Workshop on Inductive Logic Programming* (pp. 296–308). LNAI 1297, Springer.

Yamamoto, A. (2000). Using abduction for induction based on bottom generalization. In Flach, & Kakas, 2000, *Abduction and induction*: *Essays on their relation and integration. Kluwer*.

Yamamoto, A. (2002). Hypothesis finding based on upward refinement of residue hypotheses. *Theoretical Computer Science*, to appear.

Yamamoto, A., & Fronhöfer, B. (2000). Hypotheses finding via residue hypotheses with the resolution principle. In *Proceedings of the Eleventh International Conference on Algorithmic Learning Theory* (pp. 156–165). LNAI 1968, Springer.

Received April 29, 2002
Revised March 31, 2003
Accepted July 25, 2003
Final manuscript November 5, 2003