Extending Inductive Generalization with Abduction

Takashi Kanai, Susumu Kunifuji
March 29, 1999
IS-RR-99-0010

School of Information Science
Japan Advanced Institute of Science and Technology, Hokuriku
Asahidai 1–1, Tatsunokuchi
Nomi, Ishikawa, 923-12, JAPAN
kanai@jaist.ac.jp, kuni@jaist.ac.jp

ISSN 0918-7553

Abstract

We proposes an integrated framework of inductive generalization and abductive reasoning. In this framework, inductive hypotheses can be generated even if the background knowledge is not sufficient to generate hypotheses in an usual manner. The main issue of inductive generalization is to construct definitions of given examples when examples and relevant background knowledge are given. While most inductive generalization systems presuppose that the background knowledge is enough to induce definitions of given examples, the assumption is not usually met in real-world situation. In order to overcome this difficulty, we propose a framework of inductive generalization extended with abductive reasoning. This approach uses abduction when inductive generalization needs more items of background knowledge. Our approach is an integration of Inductive Logic programming and Abductive Logic Programming.

1 Introduction

The main concern of inductive generalization is to construct definitions of given examples in the situations where examples and relevant background knowledge are given. Inductive Logic Programming(ILP) studies inductive generalization in logic programming framework [Muggleton and Raedt1994]. While most inductive generalization systems assume that the given background knowledge is enough to induce definitions of given examples, the assumption is not usually satisfied in real-world.

Abductive Logic Programming(ALP) [Kakas et al.1992] is another research that deals with hypothesis in logic programming. ALP is a hypothetical reasoning framework where hypothetical language is a set of ground atoms and the representation language is a logic program. ALP has a remarkable property such that it can be viewed as an inductive inference not categorized as inductive generalization. One of the difficulties of ALP is that the explanation of a given example must be a set of ground atoms which means that ALP cannot deal with general Horn clauses. ALP therefore cannot deal with the general concept of inductive generalization.

In order to overcome the difficulty, we propose an extended framework of inductive generalization(e.g., Inductive Logic Programming) which uses the abductive reasoning (e.g., Abductive Logic Programming). We use abduction when inductive generalization requires more items of the background knowledge. In our approach items of the background knowledge are generated dynamically by abductive reasoning. In the following, we present an inductive generalization method with abduction, and compare our method with other frameworks integrating inductive generalization and abduction.

The paper is organized as follows: in Section 2 we introduce Abductive Logic Programming and Inductive Logic Programming, then discuss the difference between ALP and ILP. Section 3 describes our problem setting and outline of our approach. In Section 4 we introduce our extension of FOIL's heuristics and illustrate how our approach works. Comparison to other integration of ALP and ILP are discussed in Section 5. Section 6 concludes and presents directions for future works.

2 Background

In this section, we explain about ALP and ILP and discuss the similarities and differences between these inference schema.

2.1 Abductive Logic Programming

Abductive Logic Programming introduces the abductive reasoning into logic programming framework. Abduction is regarded as a process of finding 'missing facts', which will com-

plete the explanations of given facts. In contrast to inductive generalization, abduction can be invoked by giving only one observation.

The main objective of ALP is to find explanations that are consistent with observation. ALP finds the set of ground atoms Δ satisfying the following conditions:

- $P \not\vdash G$
- $P \cup \Delta \vdash G$
- $P \cup \Delta$ is consistent i.e. $P \cup \Delta \models I$

where P is a theory(logic program), G an observation(a goal), I an integrity constraint, Δ an explanation.

In the above formulae, the concept of *integrity constraint* is used to check the consistency of explanation under the given theory. Integrity constraints are usually used to exclude the unintended combination of missing facts, which can be thought of as contradicted hypotheses. In addition to the restriction, ALP restricts explanations to *abducibles*, a subset of pre-specified class of atoms. Abducibles are also used to exclude explanations with unintended facts.

Eshghi and Kowalski[Eshghi and Kowalski1989] propose an original abductive proof procedure for ALP in the context of the relation of Negation As Failure and abduction. Kakas and Mancarella[Kakas and Mancarella1990] extends Eshghi-Kowalski's procedure to allow for not only negative literals but also arbitrary literals as abducibles. Our system adopts a variant of Kakas and Mancarella's procedure.

2.2 Inductive Logic Programming

Inductive Logic Programming is generally regarded as the intersection of logic programming and machine learning because ILP borrows the logical theory from logic programming and adopts inductive reasoning methods from machine learning. ILP deals with the hypotheses encoded as a set of Horn clauses. ILP aims to find the rules 'generalizing examples'. The objective of ILP is defined as the process of finding the hypotheses H satisfying the following conditions:

- $B \cup E^- \not\models \square$
- $B \not\models E^+$
- $B \cup H \models E^+$
- $B \cup H \cup E^- \not\models \square$

where B is the background knowledge, $E^+(E^-)$ a set of positive(negative) examples.

One of the successful ILP systems is Quinlan's FOIL[Quinlan1990], which uses top-down search to specialize the current hypotheses incrementally. When specialization is needed, FOIL adds the most gainful literal one by one to partial clauses. Information heuristics is employed to choose the most gainful literal. In later section, we will show how heuristics employed in FOIL is extended to cope with problems due to the insufficient background knowledge.

2.3 The relation between ALP and ILP

ALP can be seen as a special case of ILP. ALP is an ILP where the hypothesis language is a set of ground atom(called abducibles) and the example is the single positive atom(called observation or goal). But the above observation is not enough to capture the difference between the two frameworks because the problem settings of ALP and ILP are both too general. In order to discover the difference between abduction and induction, we need to compare their goals. ALP differs from ILP in terms of what to learn. ILP focuses on generating clauses to characterize some given examples. The given background knowledge is also not modified in the process of induction. In ALP's view, the generated hypotheses are not considered as a characterization of the given examples, but are possible 'missing facts'. Abductive hypothesis can be seen as supplementing the given background knowledge. ILP therefore generates the knowledge(i.e., a set of Horn clauses) of the given examples, while ALP generates facts(i.e., a set of ground atoms) to supplement the given background knowledge. The goals are represented as logical consequence both in ALP and ILP.

3 Induction with Abductive Reasoning

3.1 The Problem setting

As described in the previous section, ALP and ILP have different goals i.e, ALP focuses on finding possible missing facts while ILP's goal is to characterize the given examples.

Although ALP and ILP settings are slightly different, the usual ALP and ILP settings do not capture these differences. Therefore, a common problem setting needs to integrate the two frameworks of ALP and ILP.

Since the inductive generalization is our main objective, the problem setting is a variant of problem setting for ILP extended for the ALP problem setting.

The problem setting is formalized as follows:

Given:

- The (insufficient) background knowledge P

- Abducibles Ab
- A set of integrity constraints I
- The oracle Q
- A set of positive and negative examples E
  $E^+(E^-)$ represents positive(negative) examples

Find:

A set of logic program H that can be seen as a generalization of examples, i.e, the heads of the clause in H, must be unifiable to at least one positive examples. A set of ground atoms, $\Delta \subseteq Ab$, should satisfy the following conditions:

- $P \cup E^- \not\models \square$
- $P \not\models E^+$
- $P \cup H \cup \Delta \models E^+$
- $P \cup H \cup \Delta \cup I \cup E^-$ is consistent

3.2 Abduction as Example Generation

In our approach, there are two processes, one of which is to generalize examples and the other is to find missing facts of background knowledge. Those two process use difference methods. The former uses ILP approach and the latter uses ALP approach to produce examples. Induction generalizes examples while abduction supplements the given background knowledge. Induction focuses on generating clauses to explain some given examples, while ALP generates missing facts to explain the given examples. Since we extend inductive generalization with abduction, it is natural to invoke abductive inference in the process of inductive generalization. When the insufficient part of background knowledge is needed to check whether the hypotheses imply the given examples, abductive process is dynamically used to supplement the given background knowledge. In addition, we also use our abductive procedure to check the truth of generated hypotheses.

If abductive inference is not invoked in the above situation, usual inductive techniques do not work properly because heuristics usually needs the validation of whether the given examples are explained by the current hypotheses. In the above situation, abduction is provoked to supplement the background knowledge and the generated explanations are stored as examples. This behavior can be recognized as example generation. In order to use abduction as example generation, the insufficient part of background knowledge should be declared as abducibles. The needs to supplement the insufficient part of background knowledge are recognized in advance.

4 Incorporating Abduction into top-down search

In this section, we illustrate our approach of induction with abduction.

In order to explore the hypothesis space, two typical search methods are employed in inductive generalization: top-down and bottom-up searches. In our problem setting, top-down search method is preferred to bottom-up search because the bottom-up search usually demands the sufficient background knowledge to generate appropriate hypotheses. That is, the bottom-up search firstly generates the most specific hypothesis by using the background knowledge. However the given background knowledge needs not be sufficient in our problem setting. The most specific hypothesis therefore cannot be created in our problem setting.

We will thus extend top-down search to deal with our problem setting. Our top-down search method uses FOIL-like heuristics and a greedy searching approach.

Since our approach is a variant of FOIL, the problem solving process is outlined as follows.

1. Create the most general hypothesis.
2. Until the hypothesis covers only positive examples:
   (a) Compute the gains of candidate literals like FOIL.
   In this process, abduction is invoked if background knowledge is not sufficient to prove examples. The explanations generated in this process do not presented to the user since the goal of this process is to estimate the goodness of candidate hypothesis.
   (b) Choose the most gainful literal
   In contrast to the above process, the explanations generated by the new hypothesis are shown to the user. There are two reasons why explanation is presented to the user, one reason is that the system needs to know how well the new hypothesis is. The other reason is that the explanations must be correct w.r.t. user's intention since the queried explanations are used in later induction process. If the explanations are not queried, the later induction process cannot use the correct examples.
   (c) Add the most gainful literal to current hypothesis.

4.1 Evaluation of hypotheses using abduction

Heuristics is used to estimate the goodness of hypotheses generated by inductive generalization. Heuristics that are usually employed implicitly assume that the background knowledge is sufficient to explain the given examples. If this assumption is not satisfied, the usual heuristics cannot estimate the goodness of a hypothesis because the background

knowledge is required to check whether the hypothesis implies the given examples. In fact, heuristics may lead to a wrong hypothesis in our problem setting. In addition, the insufficient part of background knowledge should not be asked to the user indiscriminately because the number of oracles becomes too many. Therefore, the heuristics should be modified to cope with the insufficient background knowledge.

In FOIL, we face the problem when choosing the most gainful literal in terms of the current hypothesis(called partial clause). Information heuristics is employed in this step to find the most gainful literal in FOIL. Literals are evaluated as follows. Let the number of + (-) bindings of a partial clause be $n^+(n^-)$. The average information that one of the bindings has label +, is

$$I(n^+, n^-) = -\log_2 n^+/(n^+ + n^-)\ bits$$

If a literal L is added to the partial clause, let k be the n^+ bindings which are not excluded by L , the numbers of bindings of the new partial clause be $m^+(m^-)$. Then the total information gained by adding L can be represented as

$$k \times (I(n^+, n^-) - I(m^+, m^-))\ bits$$

FOIL's heuristics uses the number of positive (negative) examples which are explained by a current hypothesis. However when the background knowledge is not sufficient, the number of positive(negative) examples cannot be computed correctly because the added literal L requires the incomplete part of background knowledge to be checked whether L holds or not.

To extend FOIL's heuristics we count the sum of minimum cost of examples instead of the number of positive(negative) examples. The following function is used instead of the number of positive(negative) literals.

$$n^+ = \sum_{e \in E^+} 2^{-cost(e)}$$

where $cost(x)$ is the cost function that returns the minimum cost to explain x. The $cost(x)$ equals to zero iff an explanation of x is is empty set, and $cost(x)$ is ∞ iff x cannot be explained. Other part of evaluation process remains unchanged. This evaluation is compatible to FOIL's evaluation method when e has an empty explanation set therefore this modification extends induction with abductive framework. This modification is also applied to the evaluation of negative assumptions because our abductive procedure deals with negative literals as abduction.

In the above definition, the cost of explanation (which is generated by an abductive procedure) is used to calculate the minimum cost of the explanation of given examples. ALP proof procedure does not take account of the cost of explanations. In order to represent the cost of explanation, a built-in predicate $costly/1$ is introduced. The procedural meaning

of $costly/1$ is that $costly(X)$ is true iff '$current\ cost'-X$ is less than 0. $costly/1$ is usually used for the addition of cost to the current proof in integrity constraints. For example,the integrity constraint $\leftarrow abd^*, costly(3)$ means that the abducibles abd^* cost 3.

In addition, the procedure to find the minimum cost explanation is required. Our system uses an iterative deepening search method to find the minimum cost explanation. In order to implement the search method for finding the minimum cost explanation, to represent the cost of the proof and limited cost search method is needed. We omit the detail of these algorithms because these procedures are trivial.

4.2 An Example: The lamp diagnosis problem

In this section, we show that how our system works in a situation which sufficient background knowledge is not given. Consider the following fault diagnosis problem of lamp. This problem is taken from [Ade and Denecker1995]. We modified the original problem to take cost of explanation into account.

- Let l_1, l_2, l_3, l_5 be lamps, l_4 be not a lamp.
- Let l_4 be something else, not a lamp.
- Each device $\{l_1, \ldots, l_5\}$ is connected to batteries $\{b_1, \ldots, b_5\}$.
- Each battery is empty except for l_2. It is known that the cord of l_3 is unlikely to break.

Problem: Generate the programs to explain the trouble of the lamps $l_1, \ldots, l_5$.

The following are the background knowledge and integrity constraints of this problem.

```
lamp(l1). lamp(l2). lamp(l3). lamp(l5).
conn_to_battery(l1,b1). conn_to_battery(l2,b2).
conn_to_battery(l3,b3). conn_to_battery(l4,b4).
conn_to_battery(l5,b5).
empty(b1). empty(b3). empty(b4). empty(b5).
<-- break_cord(l3),costly(3).
% every abduced atom cost 1
<-- _, costly(1).
```

Note that the abducible $break\_cord(l3)$ costs 4 because if
$break\_cord(l3)$ is assumed to be true, $costly(3)$ and $costly(1)$ are invoked simultaneously. Other abducibles (including negative literals) costs 1.

Examples can be described as following. Note that $-p(x)$ represents a negative example.

```
faulty_lamp(l1). -faulty_lamp(l2).
faulty_lamp(l3). -faulty_lamp(l4).
faulty_lamp(l5).
```

The structure of the correct programs are shown in Figure 1 . Our system first generates

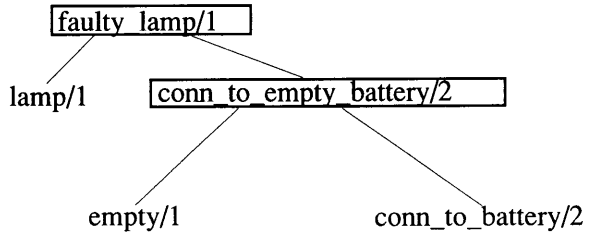

Figure 1: The structure of expected programs

the hypothesis of $failty\_lamp/1$. In the process of induction, the explanations about $conn\_empty\_battery/2$ are generated. The generated explanations are stored as examples, and then generated examples are used to generate the hypothesis about $conn\_to\_empty\_battery/2$.

The details of the behavior of our system are as follows:

Inducing $faulty\_lamp/1$

1. Make initialization and a user controls the system to induce the definition of $faulty\_lamp/1$.

2. System generates the most general hypothesis as initial hypothesis.

   clause: faulty_lamp(A) :- true

3. $lamp(A)$ is added to the partial clause because the addition of $lamp(A)$ to the partial clause excludes the negative example $faulty\_lamp(l4)$ only. Therefore the literal $lamp(A)$ is the most gainful literal.

   clause: faulty_lamp(A) :- true, lamp(A)

4. At this time, $conn\_to\_empty\_battery(A, B)$ is most gainful literal because the cost of assuming $break\_cord(l3)$ is 4, but the cost of assuming $conn\_to\_empty\_battery(l3, B)$ is 1, and the literal $empty(A)$ excludes all examples, and the gain of adding the literal $conn\_to\_battery(A, B)$ is 0(see below).

   The system selects the literal and asks for the truth values of all abductive explanations such as $conn\_to\_empty\_battery(l1, B)$ and $conn\_to\_empty\_battery(l3, B)$. The user answers, for example, $conn\_to\_empty\_battery(l1, b1)$ and $conn\_to\_empty\_battery($

$l3, b3$) are true. The system also asks for the value of $conn\_to\_empty\_battery(l2, B)$, the user indicates that $conn\_to\_empty\_battery(l2, b2)$ is false.

Then $conn\_to\_empty\_battery(A, B)$ is added to the partial clause.

clause: faulty_lamp(A) :- true, lamp(A),
conn_to_empty_battery(A, B).

In step 4, The cost of explanations are computed as follows:

- $empty(A)$

  Although the argument of the empty/1 must be battery(i.e. b1 ... b5), the variable 'A' represents lamps(i.e. l1 ... l5). Therefore this candidate excludes all positive and negative examples.

- $conn\_to\_battery(A, B)$

  Since all $l1, l2 \ldots l5$ have connection to batteries($b1, b2, \ldots b5$), there is no information gain to add this atom to hypothesis.

  $Gain = 3 * (I(3,1) - I(3,1)) = 0$

- $break\_cord(A)$

  In this candidate, the generated explanation is as follows:

  The explanation of faulty_lamp(l1) is $\{break\_cord(l1)\}$
  cost(faulty_lamp(l1)) = 1.

  The explanation of \+faulty_lamp(l2) is $\{\backslash + break\_cord(l2)\}$
  $cost(\backslash + faulty\_lamp(l2)) = 1.$

  The explanation of faulty_lamp(l3) is $\{break\_cord(l3)\}$
  $cost(faulty\_lamp(l3)) = 4.$

  The explanation of faulty_lamp(l5) is $\{break\_cord(l5)\}$
  $cost(faulty\_lamp(l4)) = 1.$

  The information gain w.r.t. $break\_cord(A)$ is:

  $$(2^{-1} * 2 + 2^{-4}) * (I(3,1) - I(2^{-1} * 2 + 2^{-4}, 2^{-1})) = 1.063 * (0.415 - 0.556)$$

  $$= -0.150$$

- $conn\_to\_empty\_battery(A, B)$

  In this candidate, the generated explanation is as follows:

The explanation of faulty_lamp(l1) is
$\{conn\_to\_empty\_battery(l1, b1)\}$
cost(faulty_lamp(l1)) = 1.

The explanation of \+faulty_lamp(l2) is
$\{\backslash + conn\_to\_empty\_battery(l2, b2)\}$
cost(faulty_lamp(l2)) = 1.

The explanation of faulty_lamp(l3) is
$\{conn\_to\_empty\_battery(l3, b3)\}$
cost(faulty_lamp(l3)) = 1.

The explanation of faulty_lamp(l5) is
$\{conn\_to\_empty\_battery(l5, b5)\}$
cost(faulty_lamp(l5)) = 1.

Therefore the information gain w.r.t. $conn\_to\_empty\_battery(A, B)$ is:

$$(2^{-1} * 3) * (I(3, 1) - I(2^{-1} * 3, 2^{-1})) = 1.5 * (0.415 - 0.415) = 0$$

There are two maximum gained literal, that are '$conn\_to\_battery(A, B)$' and '$conn\_to\_empty\_battery(A, B)$'.

In the above situation, usual FOIL system selects one of the two literals nondeterministically. In this example, the system can choose either to construct the hypothesis of faulty_lamp/1. If the system choose '$conn\_to\_battery(A, B)$', the hypothesis might be 'faulty_lamp(A) :- lamp(A), conn_to_battery(A,B), empty(B).'. Although this hypothesis is correct definition of 'faulty_lamp/1', it is not an intended definition of faulty_lamp/1. It is preferable to choose '$conn\_to\_empty\_battery(A, B)$' since the system knows the existence of the concept '$conn\_to\_empty\_battery(A, B)$'. Therefore we assume that '$conn\_to\_empty\_battery(A, B)$' is selected in this step.

Note that the system generates two different hypotheses, one is the definition of $faulty\_lamp/1$ and the other is the example of $conn\_to\_empty\_battery/2$. The generated explanations are used as examples to induce the definition of $conn\_to\_empty\_battery/2$. The following is the induction process about $conn\_to\_empty\_battery/2$.

Inducing $conn\_to\_empty\_battery/2$

1. The user indicates the system to create the definition of $conn\_to\_empty\_battery/2$
2. The system finds that $conn\_to\_empty\_battery(A, B)$ is the most gainful literal. Then the system adds the literal to the partial clause.

   clause: conn_to_empty_battery(A, B) :- true, conn_to_battery(A, B)

3. The system finds that *empty*(B) is the gainful literal. Then it adds the literal to the partial clause.

   clause: conn_to_empty_battery(A, B) :- true, conn_to_battery(A, B), empty(B).

4. The system asks the user whether or not the generated clause is correct or not.

5. The generated clause is a correct definition, and the user answers 'yes' , then all examples are explained by the generated hypotheses.

In the above process, the system creates the definition of *conn_to_empty _battery/2*. This process can be seen as producing the rule of the abduced atoms which are generated by applying generalization to the obtained explanations. Therefore, this process supplement the insufficient part of the given background knowledge.

5 Comparison to other researches

In this section, we compare our method with other integrated frameworks of ALP and ILP such as SLDNFAI, abductive concept learning, [Lamma et al.1998] and [Mooney1997].

In the following comparisons, we focus on two issues: the problem settings and the relation between abduction and induction. These issues characterize the differences between various ways for integration of induction and abduction.

SLDNFAI [Ade and Denecker1995] is an abductive and inductive procedure adopted in the abductive proof procedure called SLDNFA [Denecker and Danny1992].

The basic idea of SLDNFAI is as follows:

1. SLDNFAI first produces an explanation of the given goal. This process is same as SLDNFAI's abductive procedure. An abductive explanation(including positive and negative literal) is generated in this process.

2. Generates a set of clauses by using abductive explanations.

   SLDNFAI deal with positive abductive explanations as positive examples, and negative abductive explanations as negative examples. In this process, the methods of ILP are used to produce inductive hypothesis.

The differences between our framework and SLDNFAI are:

- Problem setting:

  There are two major differences in generating examples and the plausibility of explanation.

1. SLDNFAI firstly produces the explanation of an abductive goal, then executes their inductive generalization using generated explanation as its example. Therefore SLDNFAI first generates an abductive explanation in term of one observation, then the generated explanation is used as examples in induction process. In our approach, the abductive explanations of all examples are produced in induction process, then inductive generalization wrt. generated explanations is invoked. Therefore our approach can retain more examples than SLDNFAI approach.
2. Our approach uses a cost based abduction in the process of specializing hypotheses. SLDNFAI does not deal with the plausibility of explanation. In our approach, costs of abductive explanations are used to choose an appropriate literal which is added to the current hypothesis. This mechanism can lead to the plausible hypothesis.

- The relation between abduction and induction

  SLDNFAI calls induction procedure in abductive procedure, i.e. induction is included in abductive reasoning. In contrast to SLDNFAI, our approach executes the process of abduction in induction process, i.e. abduction is included in inductive reasoning.

Dimopoulos and Kakas propose an integrated framework of abduction and machine learning, called abductive concept learning(ACL)[Dimopoulos and Kakas1996] . In ACL, two types of input are needed, one is examples and the other is relevant observations. The term 'relevant observations' represents that the abductive explanation of given observations can supplement the background knowledge so that the given examples can be generalized. ACL first generates abductive explanations of the given observations to supplement the background knowledge, then produces hypothesis of the given examples. In later process, the abductive explanation is also treated as background knowledge.

The outline of ACL can be summarized as follows.

Input:
- E: an example of the concept to be learned
- O: a set of relevant observations

Procedure:

1. Explain the observations for each example
2. Construct the explanation sets which are not subsumed by other explanations.
3. Choose a subset of the previous explanation sets.
4. Find the conclusions under the chosen subset of explanation sets and the given background knowledge.

5. Apply an inductive generalization algorithm with the background knowledge to the conclusion generated in step (4)

In this approach, abduction and induction are invoked sequentially. Step 1-3 represent the abductive process which explains the given observations O. Step 4 and 5 express the inductive generalization using the chosen subset of explanation sets and background knowledge.

- Problem setting:

  In ACL, the input is constituted from examples and relevant observations. The ACL procedure can supplement the insufficient part of background knowledge with relevant observations. But in our problem setting, the relevant observation O is not given. Therefore, the explanations for the observations O, cannot be generated and the skeptical abductive conclusions cannot be calculated, either. In our problem setting, abductive concept learning needs to be modified to cope with the lack of observations.

- The relation between abduction and induction:

  ACL has a problem in finding relevant observations. Our approach can be viewed as searching the relevant observations of examples in the process of refining a hypothesis. Our approach specializes a hypothesis by adding the most gainful literal. The literals, which can be added to a hypothesis, are evaluated by the heuristics that can deal with the cost of explanations. This literal evaluation process is similar to the steps 1-3 in abductive concept learning. But unlike ACL, our method generates explanations from examples directly, and abduction is used when needed.

  In addition, the observations are true statements and given as input in ACL, but explanations are searched in the process of evaluating literals in our approach.

In [Lamma et al.1998], their approach can also generate the hypotheses from insufficient background knowledge. In addition, They show that their approach can generate examples from integrity constraint. The difference between [Lamma et al.1998] and our approach is that our approach can deal with the plausibility of hypotheses by extending FOIL's heuristics. But Lamma's approach does not consider this kind of plausibility, and their approach does not consider the risk that the generated explanation is not correct under the intended interpretation.

LAB algorithm[Mooney1997] also have relation to our approach. LAB and our approach use hill-climbing inductive algorithm and the accuracy(or plausibility) of generated rules is also considered in inductive process. They apply LAB algorithm to brain damage diagnosis problem and show their approach is more accurate than ID3 and PFOIL.

6 Conclusion

We proposed an integrated framework of abductive and inductive logic programming. Our approach can deal with hypotheses even if the background knowledge is incomplete. This is achieved by supplementing the insufficient background knowledge using abductive reasoning. We also extend the framework to search hypothesis space effectively by considering the cost of explanations. The heuristics is an extended FOIL's heuristics which can cope with the insufficient background knowledge when evaluating hypotheses. We have implemented a prototype of the abductive and inductive procedure in Prolog and confirmed that our system can deal with several examples including the one mentioned in the above. We compared our method with SLDNFAI and the abductive concept learning. We showed that our method can deal with the plausibility of hypotheses by considering the cost of explanation.

The similarity and difference among our method and other integrated frameworks should be studied further because each approach should be applied to different domains. Our method should also be tested against more practical tasks.

References

[Ade and Denecker1995] Hilde Ade and Marc Denecker, "AILP: Abductive Inductive Logic Programming", Proceedings of 14th International Joint Conference on Artificial Intelligence (IJCAI), 1201-1207(1995)

[Denecker and Danny1992] Marc Denecker and De Schreye Danny, "SLDNFA: An Abductive Procedure for Normal Abductive Programs", Proceedings of the Joint International Conference and Symposium on Logic Programming, 686-700(1992)

[Dimopoulos and Kakas1996] Yannis Dimopoulos and Antonis Kakas, Abduction and Learning, Advances in Inductive Logic Programming, Luc De Raedt(ed.), IOS Press(1996)

[Eshghi and Kowalski1989] K. Eshghi and Kowalski R. A., "Abduction Compared with Negation by Failure", Proceedings of the Sixth International Conference on Logic Programming, 234-254(1989)

[Kakas and Mancarella1990] A. C. Kakas and P. Mancarella, "On the relation between Truth Maintenance and Abduction, " Proceedings of 1st Pacific Rim International Conference on Artificial Intelligence,(1990).

[Kakas et al.1992] A.C. Kakas and R.A. Kowalski and F. Toni, "Abductive Logic Programming", Journal of Logic and Computation, 2-6,719-770(1992).

[Lamma et al.1998] F. Esposito, E. Lamma, D. Malerba, P. Mello, M. Milano, F. Riguzzi and G. Semeraro, "Learning abductive logic programs", ECAI'96 Workshop on Abductive and Inductive Reasoning, 1996

[Muggleton and Buntine1988] S. Muggleton and W. Buntine, "Machine invention of first-order predicates by inverting resolution", Proceedings of the Fifth International Conference on Machine Learning, 339-352(1988)

[Muggleton and Raedt1994] Stephen Muggleton and Luc de Raedt, "Inductive Logic Programming: Theory and Methods", Journal of Logic Programming,vol.19-20,629-679(1994)

[Mooney1997] Raymond J. Mooney, "Integrating Abduction and Induction in Machine Learning", Workshop on Abduction and Induction, International Joint Conference on Artificial Intelligence(1997)

[Peirce1931] Peirce, C.S., "Collected papers of Charles Sanders Peirce", Harvard University Press, (1931-1958).

[Quinlan1990] R. Quinlan, "Learning Logical Definitions from Relations", Machine Learning, vol.5, 239-266(1990)

[Shapiro1983] E. Y. Shapiro, "Algorithmic Program Debugging", MIT Press, Cambridge, MA, 1983