A Classification of Abduction :Abduction for Logic Programming

Hirata, Kouichi
Research Institute of Fundamental Information Science, Kyushu University

出版情報：RIFIS Technical Report. 77, 1993-10. Research Institute of Fundamental Information Science, Kyushu University
バージョン：
権利関係：

A Classification of Abduction : Abduction for Logic Programming

Kouichi Hirata
Research Institute of Fundamental Information Science,
Kyushu University 33, Fukuoka 812, Japan
phone: 81-92-641-1101 ext. 2329 fax: +81-92-611-2668
e-mail: hirata@rifis.kyushu-u.ac.jp

Abstract

Abduction is a methodology of scientific researches. Peirce showed three types of abduction, and expressed them by one syllogism. Recently various researches on abduction or abductive logic have been developed in the fields of automated reasoning and machine learning. In order to systematically understand such researches and to clearly discuss abduction, this paper classifies abduction into five types. This new classification is based on an interpretation of the syllogism in abduction and the definitions of hypotheses. We examine various researches on abduction so far developed and show that many researches on abduction can be placed in our classification. Furthermore, we discuss the most essential type of abduction in our classification for logic programming and default logic, and describe Prolog programs for the abduction.

1 Introduction

Charles Sanders Peirce, who was a philosopher, scientist and logician, asserted that a scientific research consists of three stages, *abduction*, *deduction*, and *induction* [Pei65, Uey79, Yon82]. In a scientific research, first we propose a hypothesis from given data. Then we derive new conclusions from the hypothesis by deduction. Finally at third stage we corroborate, by induction, the hypothesis and its conclusions. The first stage, i.e., the proposal of a hypothesis, is called *abduction*. According to Peirce, abduction is an inference which begins with an observation of *a surprising fact*, and proposes a hypothesis to explain why the fact arises. The main role of abduction is the proposal of theories. Thus abduction is *a method of scientific discovery*. An inference schema by abduction is described by the following three steps [Pei65, Uey79, Yon82].

1. A surprising fact C is observed.
2. If A were true, then C would be a matter of course.
3. Hence, there is reason to suspect that A is true.

In general, the above inference schema is depicted by a syllogism:

$$\frac{C \quad A \rightarrow C}{A}.$$

The following examples of the premises A and conclusions C in the above inference schema are found in literature:

(a) C :'these beans are white',
A :'these beans are ones which are taken out of this bag' [Pei65, Ino92];

(b) C :'I heard somebody scream at midnight',
A :'I thought she was attacked' [Uey79];

(c) C :'there is a fossil shell in a mountain',
A :'there used to be a sea' [Pei65, Yon82];

(d) C :'the Atlantic coastline in Africa and America are similar',
A :'the continental drift theory' [Uey79];

(e) C :'the evulutionary fact remaining of fossil',
A :'the theory of natural selection in biology' [Uey79];

(f) C :'the data of observations of an orbit of planets by Tycho Brahe',
A :'an orbit of planets is an oval (Kepler's first law)' [Pei65, Uey79, Yon82].

Peirce showed three types of the proposed hypothesis to explain a surprising fact [Yon82]. The first type is a hypothesis on the fact which can be confirmed, even if it is not confirmed at the abduction. The examples (a) and (b) above belong to this type. The second type is a hypothesis on the fact which physically cannot be confirmed. The example (c) belongs to this type, because we cannot confirm that *there used to be a sea*. The third type is a hypothesis on the fact which in practice and in principle cannot be confirmed by our scientific knowledge. The examples (d), (e), and (f) belong to this type, because the hypotheses A cannot be derived from the scientific knowledge they had at that time.

Peirce expressed these three types of abduction by just *one* syllogism. This is obviously unreasonable. These types of abduction should be expressed by different syllogisms, which is a point we want to make in this paper.

In philosophy of science, there has been much discussion of whether there could be a *logic of discovery*. Reichenbach [Rei38, Bro77, Tha88] proposed a sharp distinction between the *context of discovery* and the *context of justification*. He claimed that the philosophy of science should be concerned only questions of confirmation and acceptance that belong in the context of justification, and that the topic of discovery should be relegated to psychology and sociology. Furthermore, Popper [Pop59] pointed deeply that the work of the scientist consists in putting forward and testing theories. He also distinguished sharply between the process of conceiving a new idea, and the methods and results of examining it logically. In the former, there is no such thing as a logical method of having new ideas, or logical reconstruction of this process. Every discovery contains an *irrational element*. In the later, the scientific knowledge is never verified, and it is only falsified. Reichenbach and Popper adopted this sharp distinguish to eliminate psychologism, because they claimed that science should grow rationally. However, some philosophers, for example Hanson [Han58], Kuhn [Kuh70], and Brown [Bro77], have resisted this restriction. Brown [Bro77] claimed that, in a scientific discovery, the context of justification is a part of the context of discovery, and we cannot draw a line between the context of discovery and one of justification. The relation between justification and discovery has remained unclear.

According to Popper [Pop59], whether or not scientific methodology is rational is evaluated by *universally validity*. However, we do not need to preserve universally validity in scientific discovery. Therefore, we adopt abduction, which causes *fallacy of affirming the concequent*, one of the methods of scientific discovery in this paper. In fact, Hanson [Han58] pointed out that it is possible to discover scientific theories by the abduction.

In computer science, especially in computational logic and logic programming, many researchers have extensively studied the abduction from various viewpoints.

Plotkin [Plo71] has studied abduction together with inductive generalization. There exist Shapiro's model inference system [Sha81] and inductive logic programming [Mug92] as the extensions of Plotkin. These are also a sort of abduction, because they really propose hypotheses. Genest *et al.* [GMP90] and Duval citeDuv91 suggested abduction for explanation-based generalization.

Pople [Pop73, Kun87, Ino92] has given one direction for the researches of abduction. There exist Poole's Theorist [Poo88], Kunifuji's hypothesis-based reasoning [Kun87], and abductive logic programming [Dun91, EK89, KM90] as the extensions of Pople. Poole [Poo88] discussed the relationship between default logic and abduction, and showed that abduction can be viewed as a default logic. Kunifuji [Kun87] developed Poole's Theorist as a hypothesis-based reasoning system. Eshghi and Kowalski [EK89] discussed the relationship between negation as failure and abduction in logic programming. Kakas and Mancarella [KM90] and Dung [Dun91] defined an abductive framework in nonmonotonic logic programming, and studied the semantics in that framework. Out of these studies there has emerged a new field of abductive logic programming.

In order to systematically understand these various researches of abduction and clearly discuss abduction in the sense of Peirce, we classify abduction into five types. This new classification is based on an interpretation of a syllogism in Peirce's abduction and the definitions of hypotheses. We examine various researches on abduction so far developed and show that many researches on Peirce's abduction can be placed in our classification. Furthermore, we discuss the most essential type of abduction in our classification for logic programming and default logic, and describe Prolog programs for the abduction.

2 Classification of Abduction

In this section, we classify abduction for logic programming into five types.

First, we distinguish abduction of a premise and that of a theory. In abduction of a premise, a hypothesis A in a syllogism is a premise, i.e., a set of atoms. Then, for a surprising fact C and a hypothesis A, we denote abduction of a premise by $A \rightarrow C$ in a syllogism. On the other hand, in abduction of a theory, a hypothesis A in a syllogism is a theory, i.e., logic programming itself. Then, for a surprising fact C and a hypothesis A, we denote abduction of a theory by $A \vdash C$ in a syllogism.

Peirce showed three types of the explanatory hypotheses to explain a surprising fact [Yon82]. The first type is a hypothesis on the fact which can be confirmed, even if it is not confirmed at the abduction. The second type is a hypothesis on the fact which physically cannot be confirmed. The third type is a hypothesis on the fact which in practice and in principle cannot be confirmed by our scientific knowledge. In this section, we apply these three types to abduction of a premise and a theory.

For abduction of a premise, it is our purpose to obtain a rule $A \rightarrow C$ and to propose a hypothesis A. In order to obtain a rule, we apply Peirce's three types to abduction of a premise. Consequently, we suppose that the existence of rules corresponds to the confirmation in Peirce's types.

(1) The first type is abduction that assumes existence of the rules in a given program. In this case, we can select the rule from the program, and propose a premise from the surprising fact and the selected rule. We call this type of abduction *rule-selecting abduction.*

(2) The second type is abduction that assumes existence of the rules in a program other than a given one. In this case, we must find the rule in another program, and propose a premise

from the surprising fact and the found rule. We call this type *rule-finding abduction.*

(3) The third type is abduction that cannot assume existence of the rules in any program. In this case, we must newly generate the rule, and propose a premise from the surprising fact and the generated rule. We call this type *rule-generating abduction.*

According to Peirce, abduction begins with an observation of a surprising fact [Pei65, Uey79, Yon82]. Hence, in abduction of a premise, the surprising fact must be *surprising* with respect to the program given in advance. For a program P given in advance, a set of atoms A, and a surprising fact (*sf*, for short) C, the above three types of abduction are depicted by the following syllogisms:

(1) rule-selecting	$\dfrac{C\ (sf\ wrt\ P)\quad P \vdash A \rightarrow C}{P \vdash A}$
(2) rule-finding	$\dfrac{C\ (sf\ wrt\ P)\quad P' \vdash A \rightarrow C}{P \vdash A}$
(3) rule-generating	$\dfrac{C\ (sf\ wrt\ P)}{P \vdash A \rightarrow C\quad P \vdash A}$

For abduction of a theory, it is our purpose to obtain a theory A. In order to obtain a theory, we apply Peirce's three types to abduction of a theory. Consequently, we suppose that the existence of theories corresponds to the confirmation in Peirce's types.

(4) The first type is abduction that assumes existence of the theory in a given family of programs. In this case, we can select and propose a theory which makes the surprising fact true. We call this type of abduction *theory-selecting abduction.*

(5) The second type of abduction which we could call *theory-finding abduction* is the same as the theory-selecting abduction above, because we must assume existence of a family of programs.

(6) The third type is abduction that cannot assume existence of the theory in the family of programs. In this case, we generate and propose a theory which makes the surprising fact true. We call this type *rule-generating abduction.*

For a theory B given in advance and a surprising fact C, the above types of abduction are depicted by the following syllogisms:

(4) theory-selecting	$\dfrac{C\ (sf\ wrt\ B)\quad A \vdash C}{A\ (theory)}$
(6) theory-generating	$\dfrac{C\ (sf\ wrt\ B)}{A \vdash C\quad A\ (theory)}$

Now we examine the various researches on abduction so far developed and show that all of them can be fixed in our classification. Note that, in the following discussion, we never deny the researches about induction.

(1) *Rule-selecting abduction*: Abductive logic programming [Dun91, EK89, KM90] is a sort of rule-selecting abduction. It is different from Peirce's abduction in the following viewpoint: Peirce asserted that abduction begins with an observation of a surprising fact [Pei65, Uey79, Yon82]. However, in their works on abductive logic programming, Eshghi and Kowalski [EK89], Kakas and Mancarella [KM90], and Dung [Dun91] asserted that a hypothesis to explain the observed fact can be formed in the abductive framework. Kakas and Mancarella [KM90] also asserted that the abductive framework is vacuous and ill-defined if there exist no models to

explain the observation. Therefore, they cannot deal with the surprising fact in the sense of Peirce's abduction. Furthermore, abduction for explanation-based generalization by Genest *et al.* [GMP90] is also a sort of rule-selecting abduction. However, it depends on heuristics which makes the surprising fact is *surprising*.

(2) *Rule-finding abduction*: Duval's abduction is a sort of rule-finding abduction. Duval [Duv91] dealt with the following abduction for explanation-based generalization: Let D be a domain theory, $A \leftarrow B \wedge C$ be a rule in D, and C be a surprising fact of D. Then his system finds $C' \in D$ which is analogous to C, and adds a rule $A \leftarrow B \wedge C'$ to D. He called such adding rule abduction.

(3) *Rule-generating abduction*: The constructive operators such as V and W operators [Mug92] in inductive logic programming are a sort of rule-generating abduction.

(4) *Theory-selecting abduction*: Poole's Theorist [Poo88] and Kunifuji's hypothesis-based reasoning [Kun87] are theory-selecting abduction, where the candidates of a hypothesis are given in advance. The main part of their researches is how to select a suitable hypothesis from the candidates.

(6) *Theory-generating abduction*: Shapiro's model inference system [Sha81] and inductive logic programming [Mug92] is a sort of theory-generating abduction. However, model inference system and inductive logic programming make logic programs *inductively*.

In the following sections, we discuss rule-selecting abduction.

3 Rule-selecting Abduction for Logic Programming

In order to know what abduction is and to apply it to machine discovery, we need to study each type of abduction more deeply. In the following sections, we study the rule-selecting abduction, the most essential abduction, for logic programming and default logic. First we deal with the class of logic programs for which the rule-selecting abduction can be applied to propose hypotheses. We assume readers are familiar with the notions of logic programming [Llo87].

Let P be a definite program. Then we say that α is a *surprising fact* of P if $P \not\vdash \alpha$. Note here that P is given before α is given. For such an α, we propose a *hypothesis* H such that H is a set of atoms, $P \cup H \vdash \alpha$, and H is minimal with respect to set inclusion. We call the proposal of a hypothesis *abduction*. By regarding H as a conjunction of atoms and applying deduction theorem, if $P \cup H \vdash \alpha$ then $P \vdash H \rightarrow \alpha$. Hence it is a rule-selecting abduction in our classification.

Note that the above definition is the same as one in the abductive framework [Dun91, EK89, KM90, Poo88]. However, we are interested in how a hypothesis is proposed, but not in how semantics can be given in the abductive framework. The *rule-selecting abduction* for definite program is a proposal of hypotheses.

Example 1 Let us consider a definite program P such that

$$P = \left\{ \begin{array}{l} p(f(X), f(Y)) \leftarrow p(X,Y), q(X), r(Y). \\ q(f(X)) \leftarrow q(X). \\ r(f(X)) \leftarrow r(X). \\ r(a). \end{array} \right\}.$$

The least Herbrand model $M(P)$ of P is $\{r(a), r(f(a)), \cdots\}$, and an atom α with predicate p such that $P \vdash \alpha$ does not exist in $M(P)$. Hence rule-selecting abduction for P is applied in the following way: A fact $p(f(a), f(f(b)))$ is given as a surprising fact with respect to

P, i.e. $P \not\vdash p(f(a), f(f(b)))$. Then, we can select an H_i from $H_1 = \{p(f(a), f(f(b)))\}$, $H_2 = \{p(a,b), q(a), r(f(b))\}$, and $H_3 = \{p(a,b), q(a), r(b)\}$, and propose it as a hypothesis, for which it holds that $P \cup H_i \vdash p(f(a), f(f(b)))$.

The rule-selecting abduction can be realized in the following Prolog program *rs_abd*, which is a variant of partial evaluation in van Harmelen and Bundy [vHV88].

```
rs_abd(Goal,Leaves) :- clause(Goal,Clause),rs_abd(Clause,Leaves).
rs_abd((Goal1,Goal2),(Leaf1,Leaf2)) :-
        !,rs_abd(Goal1,Leaf1),rs_abd(Goal2,Leaf2).
rs_abd(Leaf,Leaf) :- !.
```

We can improve the program *rs_abd* using the idea of the *most specific abduction* in Stickel [Duv91, Ino92] as follows.

```
msrs_abd(Goal,Leaves) :- clause(Goal,Clause),msrs_abd(Clause,Leaves).
msrs_abd((Goal1,Goal2),(Leaf1,Leaf2)) :-
        !,msrs_abd(Goal1,Leaf1),msrs_abd(Goal2,Leaf2).
msrs_abd(Leaf,Leaf) :-(not clause(Leaf,X) -> true).
```

The program *msrs_abd* returns all the leaves of SLD-trees as hypotheses, while the *rs_abd* program returns all the nodes.

By using the above *rs_abd* and *msrs_abd*, we can automatically propose some hypotheses if they terminate. Hence, we can find the class of logic programs for which the programs *rs_abd* and *msrs_abd* terminate. We call the clause whose predicate symbol of the head is p a *definition clause* of p.

Let P be a definite program and p be a predicate symbol. Then a *recursive definition* of p in P, denoted by $rec(P,p)$, is a definition clause of p defined by the following procedure:

1. Select a clause in P whose head predicate is p, and let it be $rec(P,p)$.
2. For $rec(P,p) = A \leftarrow B_1, \cdots, B_l, \cdots, B_n$, if there exists a clause $E \leftarrow F_1, \cdots, F_m$ such that $B_l\theta = E\theta$ for a substitution θ, and $pred(B_l)(= pred(E)) \neq p$, then eliminate the clause $E \leftarrow F_1, \cdots, F_m$ from P, and put
$$rec(P,p) = (A \leftarrow B_1, \cdots, B_{l-1}, F_1, \cdots, F_m, B_{l+1}, \cdots, B_n)\theta.$$
3. Repeat 2 until it cannot be applied.

A *recursive program* of p in P, denoted by $RP(P,p)$, is a program consisting of $rec(P,p)$ and clauses in P which is used in constructing $rec(P,p)$.

For a definite program P and a predicate symbol p, $rec(P,p)$ and $RP(P,p)$ are not unique in general.

Example 2 Let P_1, P_2 and P_3 be the following definite programs:

$$P_1 = \left\{ \begin{array}{l} p(f(X)) \leftarrow p(X), q(X,Y). \\ q(f(X), f(Y)) \leftarrow q(f(X), Y). \end{array} \right\},$$

$$P_2 = \left\{ \begin{array}{l} p(f(X)) \leftarrow p(X), q(X,Y). \\ q(f(X), f(Y)) \leftarrow q(X, f(Y)). \end{array} \right\},$$

$$P_3 = \left\{ \begin{array}{l} p(f(X)) \leftarrow p(f(f(X)), q(X,Y). \\ q(f(X), f(Y)) \leftarrow q(f(X), Y). \end{array} \right\}$$

Then, $rec(P_i, p)$ and $RP(P_i, p)$ $(1 \leq i \leq 3)$ are:

$$rec(P_1, p) = p(f(f(X))) \leftarrow p(f(X)), q(f(X), Y).$$
$$rec(P_2, p) = p(f(f(X))) \leftarrow p(f(X)), q(X, f(Y)).$$
$$rec(P_3, p) = p(f(f(X))) \leftarrow p(f(f(f(X)))), q(f(X), Y).$$

$$RP(P_1, p) = \left\{ \begin{array}{l} p(f(f(X))) \leftarrow p(f(X)), q(f(X), Y). \\ q(f(X), f(Y)) \leftarrow q(f(X), Y). \end{array} \right\}$$

$$RP(P_2, p) = \left\{ \begin{array}{l} p(f(f(X)) \leftarrow p(f(X)), q(X, f(Y)). \\ q(f(X), f(Y)) \leftarrow q(X, f(Y)). \end{array} \right\}$$

$$RP(P_3, p) = \left\{ \begin{array}{l} p(f(f(X))) \leftarrow p(f(f(f(X)))), q(f(X), Y). \\ q(f(X), f(Y)) \leftarrow q(f(X), Y). \end{array} \right\}.$$

Let P_4 be the following definite program:

$$P_4 = \left\{ \begin{array}{l} p(f(X)) \leftarrow p(X), q(X, Y). \\ p(f(X)) \leftarrow p(f(f(X)), q(X, Y). \\ q(f(X), f(Y)) \leftarrow q(X, Y). \end{array} \right\}.$$

Then, there exist the following two $rec(P_4, p)$:

$$p(f(f(X))) \leftarrow p(f(X)), q(f(X), Y).$$
$$p(f(f(X))) \leftarrow p(f(f(f(X)))), q(f(X), Y).$$

Thus there also exist the following two $RP(P_4, p)$:

$$\left\{ \begin{array}{l} p(f(f(X))) \leftarrow p(f(X)), q(f(X), Y). \\ q(f(X), f(Y)) \leftarrow q(X, Y). \end{array} \right\},$$

$$\left\{ \begin{array}{l} p(f(f(X))) \leftarrow p(f(f(f(X)))), q(f(X), Y). \\ q(f(X), f(Y)) \leftarrow q(X, Y). \end{array} \right\}.$$

For a term t, $|t|$ denotes the length of t, that is, the number of all occurrences of symbols in t. For example, $|a|$ is 1 and $|f(f(a))|$ is 3. A clause $p(t_1, \cdots, t_n) \leftarrow B_1, \cdots, B_m$ is said to be *p-reducing* with respect to the i-th argument, if $|t_i\theta| > |s_i^l\theta|$ for any substitution θ and for any l such that $pred(B_l) = p$, where s_i^l is the i-th term of B_l. The s_i^l is called *p-reducing term*. This definition is an extension of reducing and weakly reducing programs in Yamamoto [Yam92].

Example 3 In Example 2, the definition clause of p in P_1 and P_2 are p-reducing. $rec(P_1, p)$ and $rec(P_2, p)$ are also p-reducing. The definition clause of p in P_3 and $rec(P_3, p)$ is not p-reducing.

For p-reducing clause, the following lemma holds.

Lemma 1 Let $C = p(t_1, \cdots, t_n) \leftarrow B_1, \cdots, B_m$ be a p-reducing clause and $p(s_1, \cdots, s_n)$ be a ground atom. Then all the SLD-derivations of $\{C\} \cup \{\leftarrow p(s_1, \cdots, s_n)\}$ are finite.

Proof Suppose C is p-reducing with respect to the i-th argument.

If $p(t_1, \cdots, t_n)$ and $p(s_1, \cdots, s_n)$ are not unifiable, then the derivation of $\{C\} \cup \{\leftarrow p(s_1, \cdots, s_n)\}$ is finitely failed.

Suppose $p(t_1, \cdots, t_n)$ and $p(s_1, \cdots, s_n)$ are unifiable. Since $p(s_1, \cdots, s_n)$ is ground, there exists a unifier θ for $p(t_1, \cdots, t_n)$ and $p(s_1, \cdots, s_n)$ such that $p(t_1, \cdots, t_n)\theta = p(s_1, \cdots, s_n)$.

If $B_j\theta$ is the selected atom of the goal $\leftarrow B_1\theta, \cdots, B_m\theta$, and $B_j\theta$ and $p(t_1, \cdots, t_n)$ are not unifiable, then the derivation of $\{C\} \cup \{\leftarrow B_1\theta, \cdots, B_m\theta\}$ is finitely failed. Otherwise, suppose $B_l\theta$ is the selected atom of the goal $\leftarrow B_1\theta, \cdots, B_m\theta$. Suppose $B_l\theta$ and $p(t_1, \cdots, t_n)$ are unifiable. Note that $s_i^l\theta$ is ground, where s_i^l is the i-th argument in B_l. By the definition of p-reducing clause,

$$|s_i^l\theta| < |t_i\theta| = |s_i|.$$

Furthermore, if $p(t_1, \cdots, t_n)$ and $B_l\theta$ are unifiable, then, for the derivation of $\{C\} \cup \{\leftarrow B_l\theta\}$, there exists a unifier σ for $B_l\theta$ and $p(t_1, \cdots, t_n)$ such that $B_l\theta\sigma = p(t_1, \cdots, t_n)\sigma$. Then,

$$|s_i^l\sigma| < |t_i\sigma| = |s_i^l\theta\sigma| = |s_i^l\theta| < |s_i|.$$

Hence, the longest derivation of $\{C\} \cup \{\leftarrow B_1\theta, \cdots, B_m\theta\}$ is constructed by the following way.

Let G_1 be the initial goal $\leftarrow B_1\theta, \cdots, B_m\theta$. Then, by selecting each atom $B_l\theta$ in the derivation, we can obtain the following resolvent G_m of the derivation:

$$\leftarrow (B_1\theta_1, \cdots, B_m\theta_1), (B_1\theta_2, \cdots, B_m\theta_2), \cdots, (B_1\theta_m, \cdots, B_m\theta_m).$$

For any $B_l\theta_k (1 \leq l, k \leq m)$, $|s_i^l\theta_k| < |s_i|$. Furthermore, by selecting each atom $B_l\theta_k$ in the derivation, we can also obtain the following resolvent G_{m+m^2} of the derivation:

$$\leftarrow ((B_1\theta'_1, \cdots, B_m\theta'_1), \cdots, (B_1\theta'_m, \cdots, B_m\theta'_m)), \cdots, (\cdots, (B_1\theta'_{m^2}, \cdots, B_m\theta'_{m^2})).$$

For any $B_l\theta'_k (1 \leq l \leq m, 1 \leq k \leq m^2)$, $|s_i^l\theta'_k| < |s_i| - 1$.

Hence, the length of the derivation of $\{C\} \cup \{\leftarrow B_1\theta, \cdots, B_m\theta\}$ is at most $\sum_{k=1}^{|s_i|} m^k$. The length of the derivation of $\{C\} \cup \{\leftarrow p(s_1, \cdots, s_n)\}$ is at most $1 + \sum_{k=1}^{|s_i|} m^k$. □

The condition that *rs_abd* and *msrs_abd* programs terminate on P is characterized by the following head-reducing $RP(P, p)$.

Let $rec(P, p) = p(t_1, \cdots, t_n) \leftarrow B_1, \cdots, B_m$. Then, a recursive program $RP(P, p)$ is called *head-reducing* if it satisfies the following conditions:

1. If there exists a k such that $B_k = p(s_1^k, \cdots, s_n^k)$, then
   (a) there exists a j such that $|t_j| > |s_j^k|$ for any k, and
   (b) any B_l such that $B_l = q_l(u_1^l, \cdots, u_{n_l}^l)$ $(p \neq q_l)$ satisfies one of the conditions:
      i. there exists a term u_i^l in B_l which is constructed by variables appearing in t_j, and the definition clause of q_l is q_l-reducing with respect to i-th argument, or
      ii. the definition clause of q is not included in $RP(P, p)$.
2. Otherwise, any $B_l = q_l(u_1^l, \cdots, u_{n_l}^l)$ satisfies one of the following conditions:
   (a) there exists a term u_i^l in B_l which is constructed by variables appearing in $t_1, \cdots, t_n$, and the definition clause of q_l is q_l-reducing with respect to i-th argument, or
   (b) the definition clause of q_l is not included in $RP(P, p)$.

Example 4 In Example 2, $RP(P_2, p)$ is head-reducing. On the other hand, $RP(P_1, p)$ is not head-reducing because it does not satisfy the condition 1 (b). $RP(P_3, p)$ is also not head-reducing because it does not satisfy the condition 1 (a). The first recursive program $RP(P_4, p)$ in Example 2 is head-reducing, but the second recursive program $RP(P_4, p)$ is not head-reducing. Furthermore, Prolog programs for defining a list, membership of a list and appending two lists are head-reducing.

For the termination of rule-selecting abduction, the following theorem holds.

Theorem 1 Let P be a definite program and p be a predicate symbol. If any $RP(P, p)$ is head-reducing, then all the SLD-derivations of $P \cup \{\leftarrow p(s_1, \cdots, s_n)\}$ are finite.

Proof The result is proven by induction on the number of clauses in P. If the number is 1, then Lemma 1 implies the result.

Next suppose the result is true for $P = \{C_1, \cdots, C_k\}$, and let P' be $P \cup \{C_{k+1}\}$. Any $RP(P', p)$ is head-reducing by the induction hypothesis. Let C_{k+1} be the following clause:

$$p_{k+1}(t_1, \cdots, t_{n_{k+1}}) \leftarrow B_1, \cdots, B_l.$$

If the predicate symbol p_{k+1} does not occur in P, then all the input clauses of the derivation of $P \cup \{\leftarrow p(s_1, \cdots, s_n)\}$ do not include C_{k+1}. Thus the derivation of $P' \cup \{\leftarrow p(s_1, \cdots, s_n)\}$ is equal to that of $P \cup \{\leftarrow p(s_1, \cdots, s_n)\}$. Consequently, the derivation of $P' \cup \{\leftarrow p(s_1, \cdots, s_n)\}$ is finite by the induction hypothesis.

If the predicate symbol p_{k+1} occurs in the clause C_i of P, then there exists a clause C_i in P, and one of the following cases holds:

1. p_{k+1} occurs in the body of C_i.
2. p_{k+1} occurs in the head of C_i.

If any $RP(P, p)$ does not include C_i, then all the input clauses of the derivation of $P \cup \{\leftarrow p(s_1, \cdots, s_n)\}$ do not include C_i. Thus the derivation of $P' \cup \{\leftarrow p(s_1, \cdots, s_n)\}$ is equal to that of $P \cup \{\leftarrow p(s_1, \cdots, s_n)\}$. In both cases, the derivation of $P \cup \{\leftarrow p(s_1, \cdots, s_n)\}$ is finite by the induction hypothesis.

Thus suppose some $RP(P, p)$ includes C_i.

1. Suppose p_{k+1} occurs in the body of C_i. Let C_i be the following clause:

$$p_i(u_1, \cdots, u_{n_i}) \leftarrow D_1, \cdots, D_j, \cdots, D_m \quad (pred(D_j) = p_{k+1}).$$

By the induction hypothesis, if there exists an infinite derivation of $P' \cup \{\leftarrow p(s_1, \cdots, s_n)\}$, and $D_j\lambda$ and $p_{k+1}(t_1, \cdots, t_{n_{k+1}})$ are unifiable, where λ is a substitution, then the derivation of $P' \cup \{\leftarrow D_j\lambda\}$ is infinite. However, we can show that all the derivations of $P' \cup \{\leftarrow D_j\lambda\}$ are finite by the following discussion. If the derivation of $P' \cup \{\leftarrow D_j\lambda\}$ is infinite, then the derivation of $P' \cup \{\leftarrow B_1\sigma, \cdots, B_l\sigma\}$ is also infinite, where σ is a unifier of $D_j\lambda$ and $p_{k+1}(t_1, \cdots, t_{n_{k+1}})$. For any $B_i\sigma$, consider the predicate symbol $pred(B_i)$.

(a) Suppose for any i, $pred(B_i)$ dose not occur in P or $pred(B_i)$ is p_{k+1}. Then, all the input clauses of the derivation of $P' \cup \{\leftarrow D_j\lambda\}$ include only C_{k+1}. Since $RP(P, p)$ is head-reducing, $D_j\lambda$ has the argument which is a ground term, and C_{k+1} is p_{k+1}-reducing with respect to this argument by 1.(b) i. or 2. i. in the definition of head-reducing. Then, the derivation of $P' \cup \{\leftarrow D_j\lambda\}$ is finite by Lemma 1.

(b) Suppose there exists an i such that $pred(B_i)$ occurs in P and $pred(B_i) \neq p_{k+1}$. For such i, one of the following two cases holds.

i. Suppose there exists a clause C_j such that $pred(B_i) = pred(head(C_j))$ and some $RP(P,p)$ includes C_j. If $head(C_j)$ and B_i are not unifiable, then the derivation of $P' \cup \{\leftarrow B_i\sigma\}$ is finite, because all the input clauses of the derivation of $P' \cup \{\leftarrow B_i\sigma\}$ do not include C_j. Otherwise, suppose $head(C_j)$ and B_i are unifiable. Since any $RP(P',p)$ is head-reducing and includes C_j, C_j is $pred(B_i)$-reducing with respect to some argument. Note that this argument of $B_i\sigma$ is ground. By Lemma 1, the derivation of $P' \cup \{\leftarrow B_i\sigma\}$ is finite.

ii. Suppose there exists a clause C_j such that $pred(B_i) = pred(head(C_j))$ and any $RP(P,p)$ does not include C_j. Then all the input clauses of the derivation of $P' \cup \{\leftarrow B_i\sigma\}$ do not include any clause of any $RP(P,p)$. By the case (a), the SLD-derivation of $P' \cup \{\leftarrow B_i\sigma\}$ is finite.

By (a) and (b), there exists no infinite SLD-derivation of $P' \cup \{\leftarrow B_1\sigma, \cdots, B_m\sigma\}$.

2. Suppose p_{k+1} occurs in the head of C_i. Let C_i be the following clause:

$$A \leftarrow D_1, \cdots, D_m \quad (pred(A) = p_{k+1}).$$

Let $\leftarrow G_1\tau, \cdots, G_j\tau, \cdots, G_l\tau$ be the goal whose input clause is $A \leftarrow D_1, \cdots, D_m$ for the SLD-derivation of $P \cup \{\leftarrow p(s_1, \cdots, s_n)\}$. By the induction hypothesis, if there exists an infinite SLD-derivation of $P' \cup \{\leftarrow p(s_1, \cdots, s_n)\}$, and $G_j\tau$ and A are unifiable, then the SLD-derivation of $P' \cup \{\leftarrow G_j\tau\}$ is infinite. Then the SLD-derivation of $P' \cup \{\leftarrow B_1\sigma, \cdots, B_l\sigma\}$ is infinite, where σ is a unifier of $G_j\tau$ and A. By the same proof as the case 1, there exists no infinite SLD-derivation of $P' \cup \{\leftarrow B_1\sigma, \cdots, B_l\sigma\}$.

Hence, all the SLD-derivations of $P' \cup \{\leftarrow p(s_1, \cdots, s_n)\}$ are finite. □

Corollary 1 Let P be a definite program and p be a predicate symbol. If any $RP(P,p)$ is head-reducing, then, for any ground atom $p(s_1, \cdots, s_n)$, both $rs\_abd(p(s_1, \cdots, s_n), X)$ and $msrs\_abd(p(s_1, \cdots, s_n), X)$ terminate.

4 Rule-selecting Abduction for Default Logic

In this section, we study rule-selecting abduction for default logic.

Poole [Poo88, Ino92] has defined an abductive framework and discussed its relationship to Reiter's default logic [Rei80]. We can describe the relationship in term of our abduction for logic programming in the following way:

Theorem 2 (Poole [Poo88]) Let $P \not\vdash \alpha$. Then there exists a hypothesis H such that $P \cup H \vdash \alpha$ if and only if there exists an extension E of default theory (D_H, P) such that $\alpha \in E$, where

$$D_H = \left\{ \frac{: w(X)}{w(X)} \;\middle|\; w(X) \in H \right\}.$$

The above theorem shows that there exists a belief including the explainable fact α, but it does not show how a hypothesis is constructed when a surprising fact is observed. Poole's abduction is *abduction for logic programming in term of default logic*, but not *abduction for default logic*. Now we study abduction for default logic. Here we deal with *function-free closed normal default theories whose conclusions are positive atoms*, because there exists an extension for such a theory [Rei80]. Also we deal with definite programs and the *integrity constraint* IC as negative information, where $IC = \bigvee_{i=1}^{n}(\leftarrow G_i)$ and $G_i = {G_1}^i \wedge \cdots \wedge {G_{n_i}}^i$.

Lemma 2 Let P be a definite program and IC be an integrity constraint. If $P \cup IC$ is consistent then the least Herbrand model $M(P)$ of P is the model of $P \cup IC$ under the closed-world assumption [Llo87].

In logic programming, a surprising fact is an atom α such that $P \nvdash \alpha$, and abduction is a proposal of a hypothesis H such that $P \cup H \vdash \alpha$. In default logic, a *surprising fact* is defined to be an atom α which is not included in any extension of the given default theory. For such an α, we propose a new default theory $(D, P \cup H)$, instead of the given default theory (D, P), such that some extension includes α. Hence, we regard such H as a *hypothesis*.

We transform the given default rules

$$D = \left\{ \frac{\alpha_i(\overline{X}) : w_i(\overline{X})}{w_i(\overline{X})} \;\middle|\; 1 \leq i \leq l \right\} (\overline{X} : \text{tuple of variables}),$$

to a definite program

$$P_D = \left\{ w_i(\overline{X}) \leftarrow \alpha_i(\overline{X}) \;\middle|\; \frac{\alpha_i(\overline{X}) : w_i(\overline{X})}{w_i(\overline{X})} \in D, 1 \leq i \leq l \right\}.$$

For such P_D, the following lemma holds.

Lemma 3 Let (D, P) be a closed default theory, α be a ground atom. If $P \cup P_D \nvdash \alpha$, then there exists no extensions of (D, P) which includes α.

Proof Let E be an extension of (D, P). By Reiter [Rei80], E is constructed by the following way:

$$\begin{aligned} &E_0 = P, \\ &E_{i+1} = Th(E_i) \cup \left\{ w \;\middle|\; \frac{\beta : w}{w} \in D, \beta \in E_i, \neg w \notin E \right\}, \\ &E = \bigcup_{i \geq 0} E_i. \end{aligned}$$

Since $P_D = \left\{ w \leftarrow \beta \;\middle|\; \frac{\beta : w}{w} \in D \right\}$, $E_{i+1} \subseteq Th(E_i) \cup M(P_D)$ for any i. Then, $E \subseteq M(P \cup P_D)$. Hence, if $\alpha \notin M(P \cup P_D)$ then $\alpha \notin E$. □

By the above lemma, we can regard a surprising fact in a default theory (D, P) as a fact α such that $P \cup P_D \nvdash \alpha$.

Let α be a surprising fact, $(D, P \cup IC)$ be a closed normal default theory, P be a definite program, and IC be an integrity constraint. Then, H is a *hypothesis* of the default theory $(D, P \cup IC)$ if it satisfies one of the conditions:

1. $P \cup H \vdash \alpha$ and $P \cup H \cup IC$ is consistent.
2. $P \cup H \nvdash \alpha$, $P \cup P_D \cup H \vdash \alpha$, and $P \cup P_D \cup H \cup IC$ is consistent.

Note that a hypothesis is assumed to be minimal with respect to set inclusion.

Example 5 Let (D, P) be a closed normal default theory:

$$D = \left\{ \frac{bird(X) : fly(X)}{fly(X)}, \frac{fish(X) : swim(X)}{swim(X)} \right\}, P = \left\{ \begin{array}{l} swim(X) \leftarrow penguin(X). \\ bird(X) \leftarrow penguin(X). \end{array} \right\}.$$

Then

$$P_D = \left\{ \begin{array}{l} fly(X) \leftarrow bird(X). \\ swim(X) \leftarrow fish(X). \end{array} \right\}.$$

Let IC_1 be $\leftarrow fly(X), swim(X)$.

1. If $P \cup P_D \not\vdash fly(john)$, then candidates for hypotheses are

   $$H_1 = \{fly(john)\}, H_2 = \{bird(john)\}, H_3 = \{penguin(john)\}.$$

   H_1 satisfies the first condition, and H_2 satisfies the second condition. However, H_3 does not satisfy either, because $P \cup P_D \cup H_3 \cup IC_1$ is inconsistent. Hence, H_1 and H_2 are hypotheses.

2. If $P \cup P_D \not\vdash swim(john)$, then candidates for hypotheses are

   $$H_4 = \{swim(john)\}, H_5 = \{fish(john)\}, H_6 = \{penguin(john)\}.$$

   H_4 and H_6 satisfy the first condition, and H_5 satisfies the second condition. Hence, H_4, H_5 and H_6 are all hypotheses.

Let IC_2 be $\leftarrow bird(X), swim(X)$.

1. If $P \cup P_D \not\vdash fly(john)$, then candidates of hypotheses are

   $$H_1 = \{fly(john)\}, H_2 = \{bird(john)\}, H_3 = \{penguin(john)\}.$$

   H_1 satisfies the first condition, and H_2 satisfies the second condition. However, H_3 does not satisfy either, because $P \cup P_D \cup H_3 \cup IC_1$ is inconsistent. Hence, H_1 and H_2 are hypotheses.

2. If $P \cup P_D \not\vdash swim(john)$, then candidates of hypotheses are

   $$H_4 = \{swim(john)\}, H_5 = \{fish(john)\}, H_6 = \{penguin(john)\}.$$

   H_4 satisfies the first condition, and H_5 satisfies the second condition. However, H_6 does not satisfy either, because $P \cup H_6 \cup IC_2$ is inconsistent. Hence, H_4 and H_5 are hypotheses.

Lemma 4 $M(P \cup P_D)$ is an extension of (D, P).

Proof Suppose E is constructed by the following way:

$$\begin{array}{l} E_0 = \{f \mid f \leftarrow \in P\}, \\ E_{i+1} = Th(E_i) \cup \left\{ w \;\middle|\; \frac{\alpha : w}{w} \in D, \alpha \in E_i \right\}, \\ E = \bigcup_{i \geq 0} E_i. \end{array}$$

Since $\frac{\alpha : w}{w} \in D$ is equivalent to $w \leftarrow \alpha \in P_D$, $M(P \cup P_D) = E$. By the closed-world assumption, $\neg w \notin E$ for any w. Then, E is an extension [Rei80]. □

Lemma 5 If $P \cup P_D \vdash \alpha$, then $M(P \cup P_D)$ is an extension of (D, P) which includes α.

Proof Since $P \cup P_D \vdash \alpha$, $\alpha \in M(P \cup P_D)$. By Lemma 4, $M(P \cup P_D)$ is an extension of (D, P). □

Lemma 6 If $(D, P \cup IC)$ satisfies one of the following conditions, then $M(P \cup P_D)$ is a consistent extension of $(D, P \cup IC)$ which includes α.

1. $P \vdash \alpha$, and $P \cup IC$ is consistent.
2. $P \nvdash \alpha$, $P \cup P_D \vdash \alpha$, and $P \cup P_D \cup IC$ is consistent.

Proof Suppose $(D, P \cup IC)$ satisfies the condition 1. By Lemma 2 and consistency of $P \cup IC$, $M(P)$ is the model of $P \cup IC$. Since $P \vdash \alpha$, $\alpha \in M(P)$. For any β, $P \vdash \beta$ if and only if $P \cup IC \vdash \beta$. Then, E is an extension of (D, P) if and only if E is an extension of $(D, P \cup IC)$. Since $M(P \cup P_D)$ is an extension of (D, P), $M(P \cup P_D)$ is an extension of $(D, P \cup IC)$.

Suppose $(D, P \cup IC)$ satisfies the condition 2. By Lemma 2 and consistency of $P \cup P_D \cup IC$, $M(P \cup P_D)$ is the model of $P \cup P_D \cup IC$. Since $P \cup P_D \vdash \alpha$, $\alpha \in M(P \cup P_D)$. If $P \cup IC$ is inconsistent, then $P \cup P_D \cup IC$ is also inconsistent, which contradicts the condition 2. Then, $P \cup IC$ is consistent. Hence, for any β, $P \vdash \beta$ if and only if $P \cup IC \vdash \beta$. Then, E is an extension of (D, P) if and only if E is an extension of $(D, P \cup IC)$. By Lemma 5, if $E = M(P \cup P_D)$ then $\alpha \in E$ and E is an extension of (D, P). Hence, $M(P \cup P_D)$ is an extension of $(D, P \cup IC)$.

According to Reiter [Rei80], if $P \cup IC$ is consistent, then an extension of $(D, P \cup IC)$ is also consistent. Hence, an extension of $(D, P \cup IC)$ which includes α is also consistent. □

For a closed normal default theory (D, P), the following theorem asserts that if there exists a hypothesis H satisfying the definition, then there exists an extension of $(D, P \cup H)$. Hence, we can propose a default theory in which we believe a surprising fact, when we observe it.

Theorem 3 Let (D, P) be a closed normal default theory and IC be an integrity constraint. Let $P \cup P_D \nvdash \alpha$. If there exists a hypothesis H satisfying the conditions of the definition of hypothesis, then $M(P \cup P_D \cup H)$ is a consistent extension of $(D, P \cup H \cup IC)$ which includes α.

Proof By replacing P with $P \cup H$ in the proof of Lemma 6, we can obtain the result. □

Example 6 Consider the default theory (D, P) in Example 5. In the case of IC_1, extensions E_i of $(D, P \cup H_i \cup IC_1)$ corresponding to the hypotheses H_i are:

$$E_1 = \{fly(john)\},\ E_2 = \{bird(john), fly(john)\},$$
$$E_4 = \{swim(john)\},\ E_5 = \{fish(john), swim(john)\},$$
$$E_6 = \{penguin(john), swim(john), bird(john)\}.$$

In the case of IC_2, extensions E_i of $(D, P \cup H_i \cup IC_2)$ corresponding to the hypotheses H_i are:

$$E_1 = \{fly(john)\},\ E_2 = \{bird(john), fly(john)\},$$
$$E_4 = \{swim(john)\},\ E_5 = \{fish(john), swim(john)\}.$$

5 Prolog Implementation

We describe our realization of the rule-selecting abduction for default logic as a Prolog program. The main program is *hyp*, which checks consistency under an integrity constraint and outputs hypotheses. The integrity constraint is given as the third argument of *hyp* in a form of disjunctions (expressed by ';') of conjunctions (expressed by ',') of atoms. If there exist no refutation of a goal as conjunctions of the atoms in the integrity constraint, then the hypotheses are consistent with the given program and the integrity constraint.

The predicate *hyp* works as follows: The predicate *abd* returns a hypothesis as its second argument for a input (ground goal) as its first argument. The predicate *rep_assert* adds a hypothesis given by *abd* to the original program. The predicate *consistent* checks consistency for a hypothesis and an integrity constraint. The predicate *rep_retract* removes a hypothesis added by *rep_assert* from the original program. The predicate *ic* calls an integrity constraint given as the third argument of *hyp*, and returns 'true' (*resp.* 'fail') if the integrity constraint fails (*resp.* is 'true') on the original program.

```
hyp(Goal,Hyp,IC) :-
        abd(Goal,Hyp),
        ((rep_assert(Hyp),consistent(Goal,Hyp,IC),rep_retract(Hyp))
                ->true;!,fail).
consistent(Goal,Hyp,IC) :-
        (call(Goal)->
               (call(ic(IC))->(write(': consistent'),nl) ;
               (write(': inconsistent'),nl)) ;
        (write(': inconsistent'),nl)).
ic(IC) :- (call(IC)->fail;true).
rep_assert((Atom1,Atom2)) :-
        (atom(Atom1)->assert(Atom1);rep_assert(Atom1)),
        (atom(Atom2)->assert(Atom2);rep_assert(Atom2)).
rep_assert(Atom) :- assert(Atom).
rep_retract((Atom1,Atom2)) :-
        (atom(Atom1)->retract(Atom1);rep_retract(Atom1)),
        (atom(Atom2)->retract(Atom2);rep_retract(Atom2)).
rep_retract(Atom) :- retract(Atom).
```

The termination of the above Prolog program is guaranteed by the corollary of Theorem 1:

Corollary 2 Let P be a definite program, IC be an integrity constraint, and p be a predicate symbol. For any predicate q of IC, if any $RP(P,q)$ and $RP(P,p)$ are head-reducing, then $hyp(p(s_1,\cdots,s_n),X,IC)$ terminates for any ground atom $p(s_1,\cdots,s_n)$.

When we deal with default logic in the above program, we transform the set of default rules D to a definite program P_D^+:

$$\left\{ w_i(\overline{X}) \leftarrow \alpha_i(\overline{X}), default\_w_i(\overline{X}) \;\middle|\; \frac{\alpha(\overline{X}) : w(\overline{X})}{w(\overline{X})} \in D, 1 \le i \le l \right\}.$$

Thus, we interpret $default\_w_i(\overline{t})$ appearing in a hypothesis as $w_i(\overline{t})$.

Example 7 Consider $P \cup P_D^+$ which consists of the clauses:

```
fly(X) :- bird(X),default_fly(X).     %%% default
swim(X) :- fish(X),default_swim(X).   %%% default
swim(X) :- penguin(X).                %%% theory
bird(X) :- penguin(X).                %%% theory
```

Since the above $P \cup P_D^+$ and the integrity constraint IC given in the third argument of *hyp* satisfy the conditions of Corollary 2, the *hyp* program terminates and outputs the following hypotheses:

```
: ?- hyp(fly(john),X,(fly(Y),swim(Y))).
penguin(john) , default_fly(john): inconsistent
bird(john) , default_fly(john): consistent
fly(john): consistent

: ?- hyp(swim(john),X,(fly(Y),swim(Y))).
fish(john) , default_swim(john): consistent
penguin(john): consistent
swim(john): consistent

: ?- hyp(fly(john),X,(bird(Y),swim(Y))).
penguin(john) , default_fly(john): inconsistent
bird(john) , default_fly(john): consistent
fly(john): consistent

: ?- hyp(swim(john),X,(bird(Y),swim(Y))).
fish(john) , default_swim(john): consistent
penguin(john): inconsistent
swim(john): consistent
```

6 Breadth-First Rule-Selecting Abduction

In Section 3, we introduced the class of definite programs, called head-reducing program. In this class, all derivations are finite. However, this class is strict, because there exist no local variables. For example, let P be the following program defining reversal of list:

$$\left\{ \begin{array}{l} reverse([W|X],Y) \leftarrow reverse(X,Z), concat(W,Z,Y) \\ concat(X,[W|Y],[W|Z]) \leftarrow concat(X,Y,Z) \end{array} \right\}.$$

Then, the recursive program $RP(P, reverse)$ is not head-reducing. Hence, for a surprising fact α with predicate *reverse*, rule-selecting abduction does not terminate for the program P and the goal $\leftarrow \alpha$. In this section, we introduce new abduction, called *breadth-first abduction*, whose termination is guaranteed in the above reverse programs.

The breadth-first abduction also can be realized in the following Prolog program `bfrs_abd`, where `Depth` means the given depth of the derivation.

```
bfrs_abd(Goal,Leaves,Depth) :-
        Depth > 0,clause(Goal,Clause),Depth1 is Depth-1,
        bfrs_abd(Clause,Leaves,Depth1).
bfrs_abd((Goal1,Goal2),(Leaf1,Leaf2),Depth) :-
        !,bfrs_abd(Goal1,Leaf1,Depth),bfrs_abd(Goal2,Leaf2,Depth).
bfrs_abd(Leaf,Leaf,0) :- !.
```

In order to characterize the termination of `bfrs_abd`, we introduce the condition, called *breadth-first head-reducing*, in the following way:

Definition 1 Let $rec(P,p)$ be a recursive definition $p(t_1,\cdots,t_n) \leftarrow B_1,\cdots,B_m$ of p in P. Then, the recursive program $RP(P,p)$ is *breadth-first head-reducing* if it satisfies the following conditions:

1. $pred(B_i) = p$, and there exist an atom B_i and l such that $|t_l\theta| > |s_l\theta|$.

2. $pred(B_i) \neq p$, and there exists an atom B_i such that the definition clause of $pred(B_i)$ is not included in $RP(P,p)$.

3. there exists a term s_j^k in B_k such that $|t_l\theta| > |s_j^k\theta|$, and the definition clause of $pred(B_k)$ is $pred(B_k)$-reducing with respect to the argument j.

Lemma 7 Let C be a clause $p(t_1, \cdots, t_n) \leftarrow B_1, \cdots, B_m$, and $p(s_1, \cdots, s_n)$ be a ground atom. If there exists an atom B_i which satisfies one of the following condition, then there exist a finitely failed SLD-derivation of $\{C\} \cup \{\leftarrow p(s_1, \cdots, s_n)\}$ with at most the depth $max\{|s_j| \mid 1 \leq j \leq n\}$.

1. $pred(B_i) \neq p$, or

2. $pred(B_i) = p$, and for any substitution θ, there exists a j such that $|t_j\theta| > |s_j^i\theta|$.

Proof If $\{C\} \cup \{\leftarrow p(s_1, \cdots, s_n)\}$ holds the condition 1, then this derivation is finitely failed with a depth 1 by selecting an atom B_i.

Suppose that the condition 1 does not hold. Then, for any k, $pred(B_k) = p$. If the condition 2 holds, by selecting an atom B_i which satisfies the condition 2, the resolvent

$$\leftarrow B_1\theta, \cdots, B_i\theta, \cdots, B_m\theta$$

is given by the goal $\leftarrow p(s_1, \cdots, s_n)$ and clause C. By $|t_j\theta| > |s_j^i\theta|$,

$$|s_j^i\theta| < |t_j\theta| = |s_j|.$$

Furthermore, by selecting an atom $B_i\theta$, the length of the term in the argument j is decrease 1 step by step, for this derivation of $\{C\} \cup \{\leftarrow p(s_1, \cdots, s_n)\}$. Hence, this derivation is finitely failed with at most the depth $|s_j| \leq max\{|s_j| \mid 1 \leq j \leq n\}$. □

Then, we can show the following theorem:

Theorem 4 Let α be a ground atom with predicate p. If any $RP(P,p)$ is breadth-first head-reducing, then there exists a finitely failed SLD-derivation of $P \cup \{\leftarrow \alpha\}$.

Proof Let $rec(P,p)$ be $p(t_1, \cdots, t_n) \leftarrow B_1, \cdots, B_m$. If P satisfies the condition 1 or 2 in the definition of breadth-first head-reducing, then the this theorem is trivial by Lemma 7.

Suppose that P does not satisfy condition 1 and 2, and satisfies the condition 3. Let $B_k = q(s_1^k, \cdots, s_h^k)$, and $|t_l\theta| > |s_j^k\theta|$. Let $C = q(u_1, \cdots, u_h) \leftarrow A_1, \cdots, A_{l'}$ be a definition clause of q in $RP(P,p)$. Since C is q-reducing, then, for any i such that $pred(A_i) = q$, $|u_j\theta| > |v_j\theta|$,where $A_i = q(v_1, \cdots, v_h)$. For the SLD-derivation of $P \cup \{\leftarrow \alpha\}$, the following resolvent is given:

$$\leftarrow B_1\theta, \cdots, B_k\theta, \cdots, B_m\theta.$$

Since $|s_j^k\theta| < |t_l\theta| = |s_l|$, and s_l is ground, then $s_j^k\theta$ is also ground. By selecting an atom $B_k\theta$, the resolvent

$$\leftarrow A_1\sigma, \cdots, A_{l'}\sigma$$

is given from the goal $\leftarrow B_k\theta$ and clause C. For a j-th argument v_j in $A_i\sigma$,

$$|v_j\sigma| < |u_j\sigma| = |s_j^k\theta| < |t_l\theta| = |s_l|,$$

and $v_j\sigma$ is ground. Furthermore, we can select an atom $A_i\sigma$ applied to C. Consequently, this derivation is finitely failed with at most the depth $|s_l| < max\{|s_j| \mid 1 \leq j \leq n\}$. □

7 Conclusion

We have classified abduction into five types based on an interpretation of a syllogism, examined various researches on abduction so far developed, and showed that all researches on abduction can be placed in our classification. Also we have studied the most essential type of abduction for logic programming and default logic in our classification, and have described Prolog programs for our abduction.

We can apply *hyp* program to a domain theory of explanation-based generalization [Duv91, GMP90]. Thus, if the domain theory satisfies the condition of termination, then the candidates of training examples are derived by *hyp* program.

We have left many important problems for future works. On introduction of function symbols to default logic and its program, rule-finding abduction, and rule-generating abduction we will report elsewhere.

References

[Bro77] Brown, H. I.: *Perception, theory and commitment*, Precedent Publishing, 1977.

[Dun91] Dung, P. M.: *Negation as hypothesis: an abductive foundation for logic programming*, Proceedings of 8th International Conference on Logic Programming, 3–17, 1991.

[Duv91] Duval, B.: *Abduction for explanation-based learning*, Machine Learning - EWSL 91 (LNAI 482), 348–360, 1991.

[EK89] Eshghi, K. and Kowalski, R. A.: *Abduction compared with negation by failure*, Proceedings of 6th International Conference on Logic Programming, 234–254, 1989.

[GMP90] Genest, J., Matwin, S. and Plante, B.: *Explanation-based learning with incomplete theories: a three-step approach*, Proceedings of International Workshop on Machine Learning, 286–294, 1990.

[Han58] Hanson, N. R.: *Patterns of discovery*, Cambridge University Press, 1958.

[Ino92] Inoue, K.: *Principles of abduction*, Journal of Japanese Society for Artificial Intelligence 7, 48–59, 1992 (in Japanese).

[KM90] Kakas, A. C. and Mancarella, P.: *Generalized stable models: a semantics for abduction*. Proceedings of 9th European Conference on Artificial Intelligence, 385–391, 1990.

[Kuh70] Kuhn, T. S.: *The structure of scientific revolutions*, University of Chicago Press, 1970.

[Kun87] Kunifuji, S.: *Hypothesis-based reasoning*, Journal of Japanese Society for Artificial Intelligence **2**, 22–87, 1987 (in Japanese).

[Llo87] Lloyd, J. W.: *Foundation of logic programming* (2nd edn), Springer-Verlag, 1987.

[Mug92] Muggleton, S. (ed).: *Inductive logic programming*, Academic Press, 1992.

[Pei65] Peirce, C. S.: *Collected papers of Charles Sanders Peirce (1839-1914)*, Hartshone, C. S. and Weiss, P.(eds.), The Belknap Press, 1965.

[Plo71] Plotkin, G. D.: *A further note on inductive generalization*, Machine Intelligence *6*, 1971.

[Poo88] Poole, D.: *A logical framework for default reasoning*, Artificial Intelligence *36*, 27–47, 1988.

[Pop73] Pople, Jr. H. E.: *On the mechanization of abductive logic*, Proceedings of International Joint Conference on Artificial Intelligence, 147–152, 1973.

[Pop59] Popper, K. R.: *The logic of scientific discovery*, Hutchinson, 1959.

[Rei38] Reichenbach, H.: *Experience and prediction*, University of Chicago Press, 1938.

[Rei80] Reiter, R.: *A logic for default reasoning*. Artificial Intelligence *13*, 81–132, 1980.

[Sha81] Shapiro, E. Y. *Inductive inference of theories from facts*, Research Report 192, Yale University, 1981.

[Tha88] Thagard, P.: *Computational philosophy of science*, MIT Press, 1988.

[Uey79] Ueyama, S.: *The theory of abduction*, Jinbungakuhou *43*, 103–155, 1978 (in Japanese).

[vHV88] van Harmelen, F. and Bundy, A.: *Explanation-based generalization = partial evaluation*, Artificial Intelligence *36*, 401–412, 1988.

[Yam92] Yamamoto, A.: *Procedural semantics and negative information of elementary formal system*, Journal of Logic Programming *13*, 89–97, 1992.

[Yon82] Yonemori, Y.: *Peirce's semiotics*, Keisou Syobou, 1982 (in Japanese).