Theory-Generating Abduction from Finite Good Examples

Hirata, Kouichi
Research Institute of Fundamental Information Science Kyushu University

出版情報：RIFIS Technical Report. 109, 1995-04-15. Research Institute of Fundamental Information Science, Kyushu University
バージョン：
権利関係：

RIFIS-TR-CS-109

RIFIS Technical Report

Theory-Generating Abduction from Finite Good Examples

Kouichi Hirata

April 15, 1995

Research Institute of Fundamental Information Science
Kyushu University 33
Fukuoka 812, Japan
E-mail:hirata@ai.kyutech.ac.jp Phone:0948-29-7632

Theory-Generating Abduction from Finite Good Examples

Kouichi Hirata
Department of Artificial Intelligence
Kyushu Institute of Technology
Kawazu 680-4, Iizuka 820, Japan
e-mail: hirata@ai.kyutech.ac.jp

April 15, 1995

Abstract

The theory-generating abduction is a kind of abduction which generates a theory to explain a surprising fact, and proposes it as a hypothesis. In this paper, by regarding the surprising facts as *good examples* for machine learning, we investigate theory-generating abduction from good examples. First, we introduce a subclass of logic programs, called *weakly reducing programs* $\mathcal{WR}_{\{d\}}$ *with dotted pairs*. For the class $\mathcal{WR}_{\{d\}}$, we formulate the concept of good examples, and design the algorithm of theory-generating abduction. Then, we show that the program of the class $\mathcal{WR}_{\{d\}}$ is constructed correctly by this algorithm from finite good examples. Furthermore, by using not only dotted pairs but also *concatenations*, we extend the class $\mathcal{WR}_{\{d\}}$ to *weakly reducing programs* $\mathcal{WR}_{\{d,c\}}$ *with dotted pairs and concatenations*. From the viewpoint of learning logic programs, any good example for the program of the class $\mathcal{WR}_{\{d,c\}}$ should be given by using only dotted pairs. Then, it is a problem how to determine which of the arguments' terms for any good example are described by concatenations. In this paper, we design the algorithm of theory-generating abduction for the class $\mathcal{WR}_{\{d,c\}}$, which determines such arguments, from good examples described by dotted pairs. We show that the program of the class $\mathcal{WR}_{\{d,c\}}$ is also constructed correctly by this algorithm from finite good examples described by dotted pairs.

1 Introduction

C. S. Peirce, who was a philosopher, scientist and logician, asserted that every scientific inquiry consists of three stages, *abduction*, *deduction*, and *induction* [Pei65]. According to him, every scientific inquiry begins with an observation of a *surprising fact*. The first stage, *abduction*, of scientific inquiry proposes a hypothesis to explain why the fact arises. The second stage, *deduction*, derives new conclusions from the hypotheses. The third stage, *induction*, tests empirically the hypotheses and conclusions. An inference schema of abduction is described by the following three steps [Pei65]:

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 the following syllogism:

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

Hirata [Hir93] has classified abduction in computer science into five types; *rule-selecting abduction*, *rule-finding abduction*, *rule-generating abduction*, *theory-selecting abduction*, and *theory-generating abduction*. The *theory-generating abduction*, which is the most powerful abduction for acquisition of explanatory hypotheses, is an abduction which generates a theory to explain why the surprising fact arises, and proposes it as a hypothesis. In this paper, we investigate theory-generating abduction.

According to Peirce [Pei65], abduction is an inference which begins with an observation of a *surprising fact*. By comparing theory-generating abduction with machine learning, we can regard the surprising facts for abduction as *good examples* for machine learning. The learnability from good examples has been first formulated by Freivalds *et al.* [FKW93] in the field of inductive inference of recursive functions. Furthermore, Lange *et al.* [LNW94] have applied it to language learning. On the other hand, in the field of inductive logic programming, Ling [Lin91] has investigated learning logic programs from good examples. Also Aha *et al.* [ALLM94] has discussed inductive logic programming from a small set of training examples, which are considered as good examples. In this paper, we discuss theory-generating abduction from such good examples.

As theoretical results in inductive logic programming, Shinohara [Shi90] has shown that the class of linear Prolog programs is inferable in the limit from positive examples. Arimura *et al.* [ASOI94] have extended the result by introducing some subclasses of linear Prolog. In this paper, we introduce another subclass of logic programs, called *weakly reducing programs* $\mathcal{WR}_{\{d\}}$ *with dotted pairs*, and formulate the concept of good examples for $\mathcal{WR}_{\{d\}}$. For the class $\mathcal{WR}_{\{d\}}$, we design the algorithm of theory-generating abduction from good examples. In particular, we pay our attention to the number of good examples, and show that the program of the class $\mathcal{WR}_{\{d\}}$ is constructed correctly by this algorithm from finite good examples under fixed constant symbols.

When we design logic programs, we need to use some auxiliary predicates frequently. However, in learning logic programs from examples, it is usual that no examples with the auxiliary predicates are given in a learning algorithm. Then, in the field of inductive logic programming, the problem to find such auxiliary predicates, called *predicate invention*, is one of the most important problem. In this paper, we discuss this problem by introducing a function *concatenation*, because we can easily characterize the class of logic programs by the concatenations. Then, we extend the class from $\mathcal{WR}_{\{d\}}$ to *weakly reducing programs* $\mathcal{WR}_{\{d,c\}}$ *with dotted pairs and concatenations*.

For the class $\mathcal{WR}_{\{d,c\}}$, from the viewpoint of learning logic programs, any example should be given by using only dotted pairs. Then, it is a problem of theory-generating abduction for $\mathcal{WR}_{\{d,c\}}$ how to determine which of the arguments' terms for any good example are described by concatenations. In this paper, we design the algorithm of theory-generating abduction for $\mathcal{WR}_{\{d,c\}}$, which determines such arguments, from good examples described by dotted pairs. We also show that the program of the class $\mathcal{WR}_{\{d,c\}}$ is correctly constructed by this algorithm from finite good examples described by dotted pairs under fixed constant symbols.

This paper is organized as follows: In Section 2, we prepare some notions necessary for later sections. In Section 3, we introduce the class *weakly reducing programs* $\mathcal{WR}_{\{d\}}$ *with dotted pairs*, and formulate the concept of good examples for $\mathcal{WR}_{\{d\}}$. In Section 4, we design the algorithm of theory-generating abduction for $\mathcal{WR}_{\{d\}}$. We show that the program of $\mathcal{WR}_{\{d\}}$ is constructed correctly by this algorithm from finite good examples. In Section 5, we extend the class from $\mathcal{WR}_{\{d\}}$ to *weakly reducing programs* $\mathcal{WR}_{\{d,c\}}$ *with dotted pairs and concatenations*, by introducing a function *concatenation*. The program defining the reversal of a list is included in $\mathcal{WR}_{\{d,c\}}$. In Section 6, we formulate the concept of *good examples described by dotted pairs* for $\mathcal{WR}_{\{d,c\}}$, and design the algorithm of theory-generating abduction for $\mathcal{WR}_{\{d,c\}}$. This algorithm also determines which of arguments' terms are described by concatenations. We show that the program of $\mathcal{WR}_{\{d,c\}}$ is also constructed correctly by this algorithm from finite good examples described by dotted pairs.

2 Preliminary

In logic programming [Llo87], the following T_P-operator provides the link between the declarative and procedural semantics for P. Let P be a definite program and B_P be the Herbrand base for P. Then, the mapping $T_P : 2^{B_P} \to 2^{B_P}$ is defined as follows:

$$T_P(I) = \{A \in B_P \mid A \leftarrow A_1, \cdots, A_n : \text{ a ground instance of a clause in } P,\ \{A_1, \cdots, A_n\} \subseteq I\},$$

for some Herbrand interpretation I. In order to represent the number of applications of such T_P, we define

$$T_P \uparrow 0 = \phi,$$

and, for any $n \geq 0$,

$$T_P \uparrow (n+1) = T_P(T_P \uparrow n).$$

In the following sections, we introduce the concept of good examples. In order to characterize good examples, we introduce an *unfolding operator* U_P similar to T_P-operator as follows:

$$U_P(I) = \{A\theta \mid A \leftarrow B_1, \cdots, B_n \in P,\ B_i' \in I,\ B_i\theta = B_i'\theta\ (1 \leq i \leq n)\}.$$

Also we define

$$U_P \uparrow 0 = \phi,$$

and, for any $n \geq 0$,

$$U_P \uparrow (n+1) = U_P(U_P \uparrow n) \cup U_P \uparrow n.$$

Example 1 Let P_1 be the following program:

$$P_1 = \left\{ \begin{array}{l} p(d(W,X), Y, d(W,Z)) \leftarrow p(X,Y,Z) \\ p(nil, X, X) \end{array} \right\},$$

where d means a *dotted pair*. Formally, d is a function *element* $\times$ *list* $\to$ *list* such that $d(W,X) = Y$ means that Y is a list adding W to the head of X. Then,

$U_{P_1} \uparrow 0 = \phi,$
$U_{P_1} \uparrow 1 = \{p(nil, X, X)\},$
$U_{P_1} \uparrow 2 = \{p(nil, X, X),\ p(d(W, nil), X, d(W, X))\},$
$U_{P_1} \uparrow 3 = \{p(nil, X, X),\ p(d(W, nil), X, d(W, X)),\ p(d(W_1, d(W_2, nil)), X, d(W_1, d(W_2, X))\}.$

3 Weakly Reducing Programs with Dotted Pairs

In this section, we introduce the following subclass of definite programs, called *weakly reducing programs with dotted pairs*, as the target class of theory-generating abduction.

Definition 1 Let P be the following definite programs:

$$P = \left\{ \begin{array}{l} p(t_1, \cdots, t_n) \leftarrow p(X_1, \cdots, X_n) \\ p(s_1, \cdots, s_n) \end{array} \right\}.$$

Then, P is called *weakly reducing with dotted pairs*, denoted by $P \in \mathcal{WR}_{\{d\}}$, if P satisfies the following conditions:

1. for any i $(1 \leq i \leq n)$, t_i is of the form either $d(W_i, X_i)$ or X_i, where W_i is a variable,
2. at least one j, t_j is of the form $d(W_j, X_j)$,
3. for any i $(1 \leq i \leq n)$, X_i is mutually distinct,
4. for any i, s_i is of one of the form nil, X, $d(Y, nil)$ or $d(W, Z)$,
5. if s_i is of the form either $d(Y, nil)$ or $d(W, Z)$, then t_i is of the form $d(W_i, X_i)$.

Note that $\mathcal{WR}_{\{d\}}$ is a similar class as context-free transformation $\mathcal{CFT}_{FB}^{unique}$ with a flat base, introduced by Arimura *et al.* [ASOI94].

For a ground atom α and a program $P \in \mathcal{WR}_{\{d\}}$, $input(\alpha, P)$ is the set of input clauses which is applied in the SLD-refutation of $P \cup \{\leftarrow \alpha\}$. On the other hand, $length(\alpha, P)$ is the length of the SLD-refutation of $P \cup \{\leftarrow \alpha\}$. Note that, for a program $P \in \mathcal{WR}_{\{d\}}$, the length of the SLD-refutation is determined uniquely.

Definition 2 Let $P \in \mathcal{WR}_{\{d\}}$. For a ground atom α such that $P \vdash \alpha$, α is a *good example for P* in $\mathcal{WR}_{\{d\}}$ if $input(\alpha, P) = P$, $length(\alpha, P) = 2$, and any argument's term of α includes at most one empty list *nil*.

Example 2 For P_1 in Example 1, atoms

$$\begin{array}{l} p(d(1, nil), nil, d(1, nil)), \\ p(d(1, nil), d(2, nil), d(1, d(2, nil))), \\ p(d(1, nil), d(2, d(3, nil)), d(1, d(2, d(3, nil)))), \end{array}$$

are good examples, while atoms

$$\begin{array}{l} p(nil, d(1, nil), d(1, nil)), \\ p(d(1, d(2, nil)), d(3, nil), d(1, d(2, d(3, nil)))), \\ p(d(1, d(2, nil)), d(3, d(4, nil)), d(1, d(2, d(3, d(4, nil))))), \end{array}$$

are not.

Lloyd [Llo87] pointed out that if $A \in B_P$ and $P \cup \{\leftarrow A\}$ has a refutation of length N, then $A \in T_P \uparrow N$. The following lemma claims that, if $P \in \mathcal{WR}_{\{d\}}$, then the converse also holds.

Lemma 1 Let $P \in \mathcal{WR}_{\{d\}}$. If $A \in B_P$ and $A \in T_P \uparrow N$, then $P \cup \{\leftarrow A\}$ has a refutation of at most length N.

Proof The result is proved by mathematical induction on N.

Let P be the following program:

$$P = \left\{ \begin{array}{l} p(t_1, \cdots, t_n) \leftarrow p(X_1, \cdots, X_n) \\ p(s_1, \cdots, s_n) \end{array} \right\}.$$

If $N = 1$, the result obviously holds.

Suppose that the result holds for $N - 1$. Suppose that $A \in T_P \uparrow N$. By the definition of $T_P \uparrow N$, there exists a substitution θ such that $p(t_1, \cdots, t_n)\theta = A$ and $\{p(X_1, \cdots, X_n)\theta\} \subseteq T_P \uparrow (N - 1)$. By the induction hypothesis, $P \cup \{\leftarrow p(X_1, \cdots, X_n)\theta\}$ has a refutation of at most length $N - 1$. Hence, $P \cup \{\leftarrow A\}$ has a refutation of at most length N. $\Box$

Note that Lemma 1 only depends on the form of programs. If a given program is binary, that is, the number of atoms in bodies in any clauses is at most 1, then Lemma 1 also holds for the program.

Lemma 2 Let $P \in \mathcal{WR}_{\{d\}}$. Then, α is a good example for P if and only if there exists a substitution θ such that $\alpha = \beta\theta$, where $\{\beta\} = U_P \uparrow 2 - U_P \uparrow 1$.

Proof By Lemma 1, for $A \in B_P$, $A \in T_P \uparrow N$ if and only if $P \cup \{\leftarrow A\}$ has a refutation of at most length N. Then, for any $A \in B_P$, $A \in T_P \uparrow N - T_P \uparrow (N - 1)$ if and only if $P \cup \{\leftarrow A\}$ has a refutation of just length N.

By Definition 2, α is a good example if and only if P has a refutation of the length 2, and if and only if $\alpha \in T_P \uparrow 2 - T_P \uparrow 1$. On the other hand, by the definition of T_P and U_P,

$$T_P \uparrow N = \{p(t_1, \cdots, t_n)\theta \mid p(t_1, \cdots, t_n) \in U_P \uparrow N,\ \theta : \text{ground substitution }\}.$$

Then, α is a good example for P if and only if there exists a substitution θ such that $\beta\theta = \alpha$, where $\beta \in U_P \uparrow 2 - U_P \uparrow 1$. For such β, since $P \in \mathcal{WR}_{\{d\}}$, $\{\beta\} = U_P \uparrow 2 - U_P \uparrow 1$. $\Box$

Let P and Q be formulas. Then, P is *more general* than Q, denoted by $Q \preceq P$, if there exists a substitution θ such that $P = Q\theta$. Furthermore, $P \equiv Q$ means that $P \preceq Q$ and $Q \preceq P$. In other words, P is a *variant* of Q.

For the unfolding operator, the following lemma holds:

Lemma 3 Suppose that $P_1, P_2 \in \mathcal{WR}_{\{d\}}$ and $\{\beta_k\} = U_{P_k} \uparrow 2 - U_{P_k} \uparrow 1$ $(k = 1, 2)$. Then, $\beta_1 \equiv \beta_2$ if and only if $P_1 \equiv P_2$.

Proof Suppose that $\beta_1 \equiv \beta_2$. Then, there exists an n-ary predicate symbol p such that

$$P_1 = \left\{ \begin{array}{l} p(t_1, \cdots, t_n) \leftarrow p(X_1, \cdots, X_n) \\ p(s_1, \cdots, s_n) \end{array} \right\},$$
$$P_2 = \left\{ \begin{array}{l} p(u_1, \cdots, u_n) \leftarrow p(Y_1, \cdots, Y_n) \\ p(v_1, \cdots, v_n) \end{array} \right\}.$$

By Lemma 2,

$$\beta_1 = p(t_1, \cdots, t_n)\theta, \text{ where } p(s_1, \cdots, s_n) = p(X_1, \cdots, X_n)\theta,$$
$$\beta_2 = p(u_1, \cdots, u_n)\sigma, \text{ where } p(v_1, \cdots, v_n) = p(Y_1, \cdots, Y_n)\sigma.$$

Here, $p(t_1, \cdots, t_n)\theta \equiv p(u_1, \cdots, u_n)\sigma$.

Since the variables of bodies of P_1 and P_2 are mutually distinct, we can suppose that θ and σ are of the following forms:

$$\theta = \{X_1 := s_1, \cdots, X_n := s_n\},$$
$$\sigma = \{Y_1 := v_1, \cdots, Y_n := v_n\}.$$

Suppose that $p(s_1, \cdots, s_n) \not\equiv p(v_1, \cdots, v_n)$. Then, $p(X_1, \cdots, X_n)\theta \not\equiv p(Y_1, \cdots, Y_n)\sigma$. Here, a substitution θ replaces X_i in t_i with s_i and a substitution σ replaces Y_i in u_i with v_i. Since $p(s_1, \cdots, s_n) \not\equiv p(v_1, \cdots, v_n)$, there exists an index j $(1 \leq j \leq n)$ such that $s_j \not\equiv v_j$. Then, we consider the following four cases:

1. $s_j = nil$, $v_j = d(Y, nil)$,
2. $s_j = nil$, $v_j = d(W, Z)$,
3. $s_j = d(Y, nil)$, $v_j = nil$,
4. $s_j = d(W, Z)$, $v_j = nil$.

For the case 1, t_j is of the form either X_j or $d(W_j, X_j)$, and u_j is of the form $d(V_j, Y_j)$. Then, $t_j\theta$ is of the form either nil or $d(W_j, nil)$, and $u_j\sigma$ is of the form $d(V_j, d(Y_j, nil))$. Hence, $t_j\theta \not\equiv u_j\sigma$. For the case 2, t_j is of the form either X_j or $d(W_j, X_j)$, and u_j is of the form $d(V_j, Y_j)$. Then, $t_j\theta$ is of the form either nil or $d(W_j, nil)$, and $u_j\sigma$ is of the form $d(V_j, d(W_j, Z))$. Hence, $t_j\theta \not\equiv u_j\sigma$. For the case 3 and 4, we have the same proof. Then, $\beta_1 \not\equiv \beta_2$, it is a contradiction. Hence, $p(s_1, \cdots, s_n) \equiv p(v_1, \cdots, v_n)$.

Suppose that $p(s_1, \cdots, s_n) \equiv p(v_1, \cdots, v_n)$ and $p(t_1, \cdots, t_n) \not\equiv p(u_1, \cdots, u_n)$. Since $P_1, P_2 \in \mathcal{WR}_{\{d\}}$, for any i $(1 \leq i \leq n)$, t_i is of the form either $d(W_i, X_i)$ or X_i, and u_i is of the form either $d(Z_i, Y_i)$ or Y_i. Since $p(t_1, \cdots, t_n) \not\equiv p(u_1, \cdots, u_n)$, there exists an index l $(1 \leq l \leq n)$ such that one of the following three cases holds:

$$\begin{array}{ll} t_l = d(W_l, X_l), & u_l = Y_l, \\ t_l = X_l, & u_l = d(Z_l, Y_l), \\ t_l = d(W_l, X_l), & u_l = d(Z_l, Y_l). \end{array}$$

If $t_l = d(W_l, X_l)$ and $u_l = Y_l$, then $t_l\theta = d(W_l, s_l)$ and $u_l\sigma = v_l$. Since $s_l \equiv v_l$, $t_l\theta \not\equiv v_l\sigma$. On the other hand, since $\beta_1 \equiv \beta_2$, $t_l\theta \equiv u_l\sigma$, and it is a contradiction.

If $t_l = X_l$ and $u_l = d(Z_l, Y_l)$, we obtain the same result.

If $t_l = d(W_l, X_l)$ and $u_l = d(Z_l, Y_l)$, then, since $p(t_1, \cdots, t_n) \not\equiv p(u_1, \cdots, u_n)$, there exists an index k $(1 \leq k \leq n)$ such that

$$t_k = d(W_l, X_k), \quad u_k = d(Z_k, Y_k) \ (Z_l \neq Z_k).$$

For such indices k and l, we consider the following two cases.

1. If there exists an index j such that $W_l = X_j$ and $Z_l = Y_j$, then, we obtain the following results.

$$\begin{array}{ll} t_l\theta = d(s_j, s_l), & u_l\sigma = d(v_j, u_l), \\ t_k\theta = d(s_j, s_k), & u_k\sigma = d(Z_k, v_k) \text{ or } d(v_p, v_k) \ (v_p \neq v_j). \end{array}$$

2. Otherwise, we also obtain the following results:

$$\begin{array}{ll} t_l\theta = d(W_l, s_l), & u_l\sigma = d(Z_l, u_l), \\ t_k\theta = d(W_l, s_k), & u_k\sigma = d(Z_k, v_k) \ (Z_k \neq Z_l). \end{array}$$

Hence, $\beta_1 = p(t_1, \cdots, t_n)\theta \not\equiv p(u_1, \cdots, u_n)\sigma = \beta_2$, and it is a contradiction.

The converse obviously holds. □

4 Theory-Generating Abduction for $\mathcal{WR}_{\{d\}}$

The *theory-generating abduction* is a kind of abduction which generates a theory to explain a surprising fact and proposes it as a hypothesis. An inference schema of theory-generating abduction is depicted as follows:

$$\frac{C\text{: } \textit{surprising fact} \text{ wrt } B}{\begin{array}{l}\textit{Generate} \text{ a theory } A \text{ such that } A \text{ makes } C \text{ true} \\ \textit{Propose} \text{ a hypothesis } A\end{array}}.$$

In this paper, we deal with the simplest theory-generating abduction such that $B = \phi$. Hence, the above inference schema is described by the following syllogism:

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

In this section, we design the algorithm of theory-generating abduction for $\mathcal{WR}_{\{d\}}$. First, we introduce some notions and lemmas.

For a term t_i, the length $|t_i|$ of t_i is defined as follows: $|t_i| = 1$ if $t_i = nil$, and $|t_i| = l+1$ if $t_i = d(a, s_i)$ and $|s_i| = l$. For an atom $\alpha = p(t_1, \cdots, t_n)$, the length $|\alpha|$ of α is defined as $|t_1| + \cdots + |t_n|$.

By considering the algorithm *PROPOSE_*$\mathcal{WR}_{\{d\}}$ as Figure 1, we obtain the following lemma. The algorithm *PROPOSE_*$\mathcal{WR}_{\{d\}}$ is a refined version of the algorithm *PROPOSE* for rule-generating abduction [Hir94].

Lemma 4 Suppose that $P \in \mathcal{WR}_{\{d\}}$ and $\{\beta\} = U_P \uparrow 2 - U_P \uparrow 1$. Then, P is constructed correctly from β in $O(|\beta|)$ time.

Proof Let P be an output of *PROPOSE_*$\mathcal{WR}_{\{d\}}(\beta, P)$. Suppose that P is of the following form:

$$P = \left\{ \begin{array}{l} head \leftarrow body \\ (body)\theta \end{array} \right\},$$

where $\beta = (head)\theta$. Then, $U_P \uparrow 0 = \phi$, $U_P \uparrow 1 = \{(body)\theta\}$, and $U_P \uparrow 2 = \{(body)\theta, (head)\theta\}$. Hence, $U_P \uparrow 2 - U_P \uparrow 1 = \{(head)\theta\} = \{\beta\}$. By Lemma 3, β characterizes uniquely P. Hence, P is constructed correctly from β.

It is obvious that the algorithm *PROPOSE_*$\mathcal{WR}_{\{d\}}$ runs in $O(|\beta|)$ time. □

By using Plotkin's *least general generalization lgg* [Plo70], the following lemma holds.

Lemma 5 If α_1 and α_2 are good example for P, then there exists a substitution θ such that the least general generalization $lgg(\alpha_1, \alpha_2)$ of α_1 and α_2 is equal to $\beta\theta$, where $\{\beta\} = U_P \uparrow 2 - U_P \uparrow 1$.

Proof By Lemma 2, β is a common generalization of α_1 and α_2. Then, β is also a generalization of $lgg(\alpha_1, \alpha_2)$. □

Theorem 1 Suppose that $P \in \mathcal{WR}_{\{d\}}$, $\{\beta\} = U_P \uparrow 2 - U_P \uparrow 1$, and $\{\alpha_i\}_{i \in N}$ is a family of good examples for P. Then, there exists an index l such that $lgg(\alpha_1, \cdots, \alpha_l) \equiv \beta$.

Proof Let β be an atom $p(u_1, \cdots, u_n)$ and γ_i be the following atom defined inductively:

$$\begin{array}{l}\gamma_1 = \alpha_1, \\ \gamma_{i+1} = lgg(\gamma_i, \alpha_{i+1}) \ (i \geq 1).\end{array}$$

```
Algorithm $PROPOSE\_\mathcal{WR}_{\{d\}}(\beta, \{C_1, C_2\})$
input $\beta$ : atom
output $P = \{C_1, C_2\} \in \mathcal{WR}_{\{d\}}$
for $i = 1$ to $n$
    if $t_i$ is of the form $d(W_1^i, \cdots)$ then
        $head\_arg_i := d(W_1^i, X_i)$;            /* $X_i$ is a new variable */
    else
        $head\_arg_i := X_i$;
    end if
end for
$head := p(head\_arg_1, \cdots, head\_arg_n)$;
$body := p(X_1, \cdots, X_n)$;
$C_1 := head \leftarrow body$;
$C_2 := (body)\theta$, where $\beta = (head)\theta$;
output $P = \{C_1, C_2\}$
```

Figure 1: Algorithm $PROPOSE\_\mathcal{WR}_{\{d\}}$

Let P be the following program:

$$P = \left\{ \begin{array}{l} p(t_1, \cdots, t_n) \leftarrow p(X_1, \cdots, X_n) \\ p(s_1, \cdots, s_n) \end{array} \right\}.$$

By the definition of β, for any i, u_i is of the form nil, X, $d(Y, nil)$, $d(W, Z)$, $d(U, d(V, nil))$, or $d(Q, d(R, S))$. By Lemma 5, $\gamma_l \preceq \beta$ and $\gamma_l \preceq \gamma_{l+1}$ for any l.

Let γ_j and α_j be atoms $p(v_1^j, \cdots, v_n^j)$ and $p(w_1^j, \cdots, w_n^j)$. Since $P \in \mathcal{WR}_{\{d\}}$, P is independent of the constant symbols appearing in γ_j. Then, there exists an index j such that γ_j includes no constant symbols except an empty list nil. Then, it is sufficient to prove the case that v_i^j is of the form $d(A_1, \cdots, d(A_m, nil) \cdots)$ for some j and u_i is of the form X, $d(W, Z)$ or $d(Q, d(R, S))$.

1. If $t_i = X_i$ and $s_i = X$, then, by the definition of good examples, there exists an index $k (\geq j)$ such that $w_i^k = nil$. For this index k, $v_i^{k+1} = X = u_i$.

2. If $t_i = d(W_i, X_i)$ and $s_i = X$, then, by the definition of good examples, there exists an index $k (\geq j)$ such that $w_i^k = d(a_1, nil)$. For this index k, $v_i^{k+1} = d(W, Z) = u_i$.

3. If $t_i = d(W_i, X_i)$ and $s_i = d(W, nil)$, then, by the definition of good examples, the length of any good examples is just 2. Then, there exists an index $k (\geq j)$ such that $v_i^k = d(T, d(U, nil)) = u_i$.

4. If $t_i = d(W_i, X_i)$ and $s_i = d(W, X)$, then, by the definition of good examples, there exists an index $k (\geq j)$ such that $w_i^k = d(a_1, d(a_2, nil))$. For this index k, $v_i^{k+1} = d(Q, d(R, S)) = u_i$.

For each $i (1 \leq i \leq n)$, let k_i be the maximal index satisfying the above cases and l be $max\{k_1, \cdots, k_n\}$. Then, $lgg(\alpha_1, \cdots, \alpha_l) = \beta$. □

Theorem 1 claims that the program $P \in \mathcal{WR}_{\{d\}}$ is constructed correctly *in the limit* from good examples for P.

Let Σ be a finite set, called an *alphabet*. For a program P, Σ means the set of all constant symbols appearing in P. Then, the following lemma holds:

Lemma 6 Let Σ be an alphabet $\{a_1, \cdots, a_m, nil\}$. For $\mathcal{WR}_{\{d\}}$, the number of ground atoms of the form $p(t_1, \cdots, t_n)$ under Σ such that $|t_i| \leq 4$ and t_i includes at most one empty list *nil* for any i $(1 \leq i \leq n)$ is at most $(1 + m + m^2 + m^3)^n$.

Proof The form of t_i is nil, $d(a_i, nil)$, $d(a_i, d(a_j, nil))$, or $d(a_i, d(a_j, d(a_k, nil)))$. Then, the number of the the selection of t_i is at most $1 + m + m^2 + m^3$. Hence, the number of ground atoms under Σ is at most $(1 + m + m^2 + m^3)^n$. □

By Lemma 6, the following theorem also holds:

Theorem 2 Let $\Sigma = \{a_1, a_2, a_3, nil\}$ be an alphabet. Suppose that $P \in \mathcal{WR}_{\{d\}}$ and $\{\beta\} = U_P \uparrow 2 - U_P \uparrow 1$. Then, for the set G of all good examples $p(t_1, \cdots, t_n)$ for P under Σ such that $|t_i| \leq 4$ for any i $(1 \leq i \leq n)$, $lgg(G) \equiv \beta$.

Proof Suppose that $p(t_1, \cdots, t_n)$ is a good example for P in $\mathcal{WR}_{\{d\}}$. By Definition 2, t_i includes at most one empty list *nil* for any i $(1 \leq i \leq n)$. By Lemma 6, the number of good examples of the form $p(t_1, \cdots, t_n)$ such that $|t_i| \leq 4$ for any i $(1 \leq i \leq n)$ is finite. Let k be such the number.

Suppose that $\{\beta\} = U_P \uparrow 2 - U_P \uparrow 1$ and $\beta = p(u_1, \cdots, u_n)$. Then, u_i is of the form nil, X, $d(Y, nil)$, $d(W, Z)$, $d(U, d(V, nil))$, or $d(R, d(S, T))$. Let G be the set of all good examples for P under Σ such that $|t_i| \leq 4$ for any i $(1 \leq i \leq n)$. For any good example $p(t_1, \cdots, t_n) \in S$, t_i is of the form nil, a_i, $d(a_i, nil)$, $d(a_i, d(a_j, nil))$ or $d(a_i, d(a_j, d(a_k, nil)))$.

For any good example $p(t_1, \cdots, t_n) \in S$, if t_i is of the form nil, then u_i is also of the form nil. If t_i is of the form a_i, then u_i is of the form X. If t_i is of the form $d(a_i, nil)$, then u_i is of the form $d(Y, nil)$. If t_i is of the form $d(a_i, d(a_j, nil))$, then u_i is of the form $d(U, d(V, nil))$. If t_i is of the form $d(a_i, nil)$, $d(a_i, d(a_j, nil))$ or $d(a_i, d(a_j, d(a_k, nil)))$, then u_i is of the form $d(W, Z)$. If t_i is of the form $d(a_i, d(a_j, nil))$ or $d(a_i, d(a_j, d(a_k, nil)))$, then u_i is of the form $d(Q, d(R, S))$. If there exists an index i such that $t_i = nil$, then u_i is of the form X. By the definition of $\mathcal{WR}_{\{d\}}$, there exists no case such that good examples are of the form either $d(a_i, nil)$ or $d(a_i, d(a_j, d(a_k, nil)))$. Also there exists no case such that good examples are of the form either nil or $d(a_i, d(a_j, nil))$.

Furthermore, let l' be l in Theorem 1, under the condition such that $\Sigma = \{a_1, a_2, a_3, nil\}$. Since k is the number of all good examples under Σ, then $l' \leq k$. Hence, by Theorem 1 and the above proof, the form of u_i is determined by all good examples of the form $p(t_1, \cdots, t_n)$ under Σ such that $|t_i| \leq 4$ for any i $(1 \leq i \leq n)$. □

In other words, by using the algorithm $FIN\_CST\_\mathcal{WR}_{\{d\}}(G, P)$ of theory-generating abduction for $\mathcal{WR}_{\{d\}}$ as Figure 1, the program $P \in \mathcal{WR}_{\{d\}}$ is constructed correctly from *finite* good examples G.

Example 3 Let an alphabet Σ be a set $\{1, 2, 3, nil\}$.

1. Suppose that the ground atoms $p(d(1, nil))$, $p(d(2, nil))$, and $p(d(3, nil))$ are given as good examples for P_2. Then, we obtain the atom $p(d(W, nil))$ as the least general generalizations of good examples. By $PROPOSE\_\mathcal{WR}_{\{d\}}$, we also obtain the following program P_2:

$$P_2 = \left\{ \begin{array}{l} p(d(W, X_1)) \leftarrow p(X_1) \\ p(nil) \end{array} \right\}.$$

   The program P_2 means that the argument's term of p is a list.

```
Algorithm $FIN\_CST\_\mathcal{WR}_{\{d\}}(G, P)$
input $G$ : the set of all good examples for $P$
            under $\{a_1, a_2, a_3, nil\}$ such that $|t_i| \leq 4$ for any $i$ $(1 \leq i \leq n)$
output $P \in \mathcal{WR}_{\{d\}}$
read $G$;
$\gamma := lgg(G)$;
$PROPOSE\_\mathcal{WR}_{\{d\}}(\gamma, P)$;
output $P$
```

Figure 2: Algorithm $FIN\_CST\_\mathcal{WR}_{\{d\}}$

2. Suppose that the following ground atoms are given as good examples for P_3:

$$\begin{array}{lll} p(d(1,d(1,nil))), & p(d(1,d(2,nil))), & p(d(1,d(3,nil))), \\ p(d(2,d(1,nil))), & p(d(2,d(2,nil))), & p(d(2,d(3,nil))), \\ p(d(3,d(1,nil))), & p(d(3,d(2,nil))), & p(d(3,d(3,nil))). \end{array}$$

Then, we obtain the atom $p(d(W_1, d(W_2, nil)))$ as the least general generalizations of good examples. By $PROPOSE\_\mathcal{WR}_{\{d\}}$, we also obtain the following program P_3:

$$P_3 = \left\{ \begin{array}{l} p(d(W_1, X_1)) \leftarrow p(X_1) \\ p(d(W_2, nil)) \end{array} \right\}.$$

The program P_3 means that the argument's term of p is a list of at least length 1.

Since $M_{P_3} \subset M_{P_2}$ for the least Herbrand model M_{P_2} and M_{P_3} of P_2 and P_3, $\mathcal{WR}_{\{d\}}$ is not finitely inferable from all positive examples (*c.f.* Mukouchi [Muk92]), while $\mathcal{WR}_{\{d\}}$ is finitely inferable from good examples under Σ such that the length of all the arguments' terms is at most 4.

Example 4 Let Σ be an alphabet $\{1, 2, 3, nil\}$. Suppose that the following ground atoms are given as good examples for P_4:

$$\begin{array}{lll} p(1,d(1,d(1,nil))), & p(2,d(1,d(2,nil))), & p(3,d(1,d(3,nil))), \\ p(1,d(2,d(1,nil))), & p(2,d(2,d(2,nil))), & p(3,d(2,d(3,nil))), \\ p(1,d(3,d(1,nil))), & p(2,d(3,d(2,nil))), & p(3,d(3,d(3,nil))), \\ p(1,d(1,d(1,d(1,nil)))), & p(2,d(1,d(2,d(1,nil)))), & p(3,d(1,d(3,d(1,nil)))), \\ p(1,d(1,d(1,d(2,nil)))), & p(2,d(1,d(2,d(2,nil)))), & p(3,d(1,d(3,d(2,nil)))), \\ p(1,d(1,d(1,d(3,nil)))), & p(2,d(1,d(2,d(3,nil)))), & p(3,d(1,d(3,d(3,nil)))), \\ p(1,d(2,d(1,d(1,nil)))), & p(2,d(2,d(2,d(1,nil)))), & p(3,d(2,d(3,d(1,nil)))), \\ p(1,d(2,d(1,d(2,nil)))), & p(2,d(2,d(2,d(2,nil)))), & p(3,d(2,d(3,d(2,nil)))), \\ p(1,d(2,d(1,d(3,nil)))), & p(2,d(2,d(2,d(3,nil)))), & p(3,d(2,d(3,d(3,nil)))), \\ p(1,d(3,d(1,d(1,nil)))), & p(2,d(3,d(2,d(1,nil)))), & p(3,d(3,d(3,d(1,nil)))), \\ p(1,d(3,d(1,d(2,nil)))), & p(2,d(3,d(2,d(2,nil)))), & p(3,d(3,d(3,d(2,nil)))), \\ p(1,d(3,d(1,d(3,nil)))), & p(2,d(3,d(2,d(3,nil)))), & p(3,d(3,d(3,d(3,nil)))). \end{array}$$

Then, we obtain the atom $p(X, d(W, d(X, Y)))$ as the least general generalization of the above good examples. By $PROPOSE\_\mathcal{WR}_{\{d\}}$, we also obtain the following program:

$$P_4 = \left\{ \begin{array}{l} p(X_1, d(W, X_2)) \leftarrow p(X_1, X_2) \\ p(X, d(X, Y)) \end{array} \right\}.$$

5 Introducing Function as Auxiliary Predicate

Consider the following program P_{rev} defining reversal of list:

$$P_{rev} = \left\{ \begin{array}{l} rev(d(W,X),Y) \leftarrow rev(X,Z), con(W,Z,Y) \\ rev(nil, nil) \\ con(X,d(W,Y),d(W,Z)) \leftarrow con(X,Y,Z) \\ con(X,nil,d(X,nil)) \end{array} \right\}.$$

The program P_{rev} means that the second argument's list of *rev* is the reversal of the first argument's list. The least Herbrand model $M_{P_{rev}}|_{rev}$ with the predicate symbol *rev* is as follows:

$$\{rev(nil,nil), rev(d(a,nil),d(a,nil)), rev(d(a,d(b,nil)),d(b,d(a,nil))),\cdots\}.$$

Then, the following proposition holds:

Proposition 1 There exists no program $P \in \mathcal{WR}_{\{d\}}$ such that $M_P = M_{P_{rev}}|_{rev}$.

Proof Suppose that there exists a program $P \in \mathcal{WR}_{\{d\}}$ such that $M_P = M_{P_{rev}}|_{rev}$. Then,

$$P = \left\{ \begin{array}{l} C_1 : rev(t_1,t_2) \leftarrow rev(X_1,X_2) \\ C_2 : rev(s_1,s_2) \end{array} \right\}.$$

For positive examples of the program defining reversal of list, the length of the first argument or p is equal to one of the second argument. Then, C_2 is of the form either $C_{21} : rev(nil,nil)$ or $C_{22} : revp(X,X)$. Furthermore, C_1 is one of the following forms:

$$\begin{array}{l} C_{11} : rev(d(W_1,X_1),d(W_2,X_2)) \leftarrow rev(X_1,X_2), \\ C_{12} : rev(d(W,X_1),d(W,X_2)) \leftarrow rev(X_1,X_2), \\ C_{13} : rev(X_1,d(W,X_2)) \leftarrow rev(X_1,X_2), \\ C_{14} : rev(d(W,X_1),X_2) \leftarrow rev(X_1,X_2). \end{array}$$

1. If $P = \{C_{11}, C_{21}\}$ or $\{C_{12}, C_{21}\}$, then $P \models rev(d(a,d(b,nil)),d(a,d(b,nil)))$, but $rev(d(a,d(b,nil)),d(a,d(b,nil))) \notin M_{P_{rev}}|_{rev}$.
2. If $P = \{C_{13}, C_{21}\}$, then $P \models rev(nil,d(a,nil))$, but $rev(nil,d(a,nil)) \notin M_{P_{rev}}|_{rev}$.
3. If $P = \{C_{14}, C_{21}\}$, then $P \models rev(d(a,nil),nil)$, but $rev(d(a,nil),nil) \notin M_{P_{rev}}|_{rev}$.

For the above four programs, by using C_{22} instead of C_{21}, we have the same proof. Hence, there exists no program $P \in \mathcal{WR}_{\{d\}}$ such that $M_P = M_{P_{rev}}|_{rev}$. $\square$

If we deal with only dotted pairs as function symbols, then we need to invent predicate symbols such as *con* in P_{rev} in order to design the program defining reversal of list. Hence, we introduce another function c : *element* $\times$ *list* $\rightarrow$ *list*, called a *concatenation*, such that $c(W,X) = Y$ means that Y is a list adding W to the last of X. Then, we can obtain the following programs defining reversal of list:

$$P^1_{rev} = \left\{ \begin{array}{l} p(d(W,X),c(W,Y)) \leftarrow p(X,Y) \\ p(nil,nil) \end{array} \right\},$$

$$P^2_{rev} = \left\{ \begin{array}{l} p(c(W,X),d(W,Y)) \leftarrow p(X,Y) \\ p(nil,nil) \end{array} \right\}.$$

By using dotted pairs d and concatenations c, we introduce the following subclass of definite programs.

Definition 3 Let P be the following definite programs:

$$P = \left\{ \begin{array}{l} p(t_1, \cdots, t_n) \leftarrow p(X_1, \cdots, X_n) \\ p(s_1, \cdots, s_n) \end{array} \right\}.$$

Then, P is called *weakly reducing with dotted pairs and concatenations*, denoted by $P \in \mathcal{WR}_{\{d,c\}}$, if P satisfies the following conditions:

1. for any i $(1 \leq i \leq n)$, t_i is of one of the form $d(W_i, X_i)$, $c(W_i, X_i)$ or X_i, where W_i is a variable,
2. at least one j, t_j is of the form either $d(W_j, X_j)$ or $c(W_j, X_j)$,
3. for any i $(1 \leq i \leq n)$, X_i is mutually distinct,
4. for any i, s_i is of one of the form nil, X, $d(Y, nil)$, $d(W, Z)$ or $c(W, Z)$,
5. if s_i is of the form either $d(Y, nil)$ or $d(W, Z)$, then t_i is of the form $d(W_i, X_i)$,
6. if s_i is of the form either $c(Y, nil)$ or $c(W, Z)$, then t_i is of the form $c(W_i, X_i)$.

Obviously, both P^1_{rev} and P^2_{rev} are included in $\mathcal{WR}_{\{d,c\}}$.

6 Theory-Generating Abduction for $\mathcal{WR}_{\{d,c\}}$

In this section, we investigate theory-generating abduction for $\mathcal{WR}_{\{d,c\}}$ from good examples. If any good example is given by using dotted pairs and concatenations according to an intended program, then we can construct the program of $\mathcal{WR}_{\{d,c\}}$ as similar as Lemma 6 and Theorem 2, by using for-loop of *PROPOSE_*$\mathcal{WR}_{\{d,c\}}$ instead of one of *PROPOSE_*$\mathcal{WR}_{\{d\}}$.

On the other hand, from the viewpoint of learning logic programs from examples, any example should be given by using only dotted pairs. Then, in this section, we discuss the problem of theory-generating abduction for $\mathcal{WR}_{\{d,c\}}$ how to introduce a function concatenation into examples described by dotted pairs.

In order to solve this problem, we prepare two operators *dot* and *con* as follows: For a term t, $con(t)$ (*resp.*, $dot(t)$) is an equivalence term described by concatenations (*resp.*, dotted pars) if t is described by dotted pairs (*resp.*, concatenations); otherwise $con(t) = t$ (*resp.*, $dot(t) = t$). For example, $con(d(1, d(2, d(3, nil)))) = c(3, c(2, c(1, nil)))$, and $dot(c(1, c(2, c(3, nil)))) = d(3, d(2, d(1, nil)))$. For an atom $p(t_1, \cdots, t_n)$, $dot(p(t_1, \cdots, t_n)) = p(dot(t_1), \cdots, dot(t_n))$. For a set S of atoms $\{\alpha_1, \cdots, \alpha_k\}$, $dot(S) = \{dot(\alpha_1), \cdots, dot(\alpha_k)\}$.

Let α be a ground atom $p(t_1, \cdots, t_n)$ described by only dotted pairs. For any j $(1 \leq j \leq 2^n)$, we define α^j as follows:

$$\begin{array}{ll} \alpha^1 & = p(t_1, t_2, \cdots, t_n) = \alpha, \\ \alpha^2 & = p(con(t_1), t_2, \cdots, t_n), \\ \alpha^3 & = p(t_1, con(t_2), \cdots, t_n), \\ & \cdots\cdots \\ \alpha^{2^n} & = p(con(t_1), con(t_2), \cdots, con(t_n)). \end{array}$$

Note that $dot(\alpha^j) = \alpha$ for any j $(1 \leq j \leq 2^n)$.

Since we consider that any good example should be described by only dotted pairs, then we introduce the following definition. Note that, for a dotted pair d and a concatenation c, $d(X, nil) = c(X, nil)$, but $d(W_1, d(W_2, nil)) \neq c(W_1, c(W_2, nil))$. Then, $length(\alpha, P) = 3$ in the following definition.

Definition 4 Let $P \in \mathcal{WR}_{\{d,c\}}$ and $\alpha_i = p(t_1^i, \cdots, t_n^i)$ $(1 \leq i \leq k)$. Then, $\{\alpha_1, \alpha_2, \cdots, \alpha_k\}$ is the set of *good examples for P described by dotted pairs* if there exists an index j $(1 \leq j \leq 2^n)$ such that, for any i $(1 \leq i \leq k)$, $\alpha_i = \alpha_i^1$ and α_i^j satisfies the following conditions:

1. $P \vdash \alpha_i^j$,
2. $input(\alpha_i^j, P) = P$,
3. $length(\alpha_i^j, P) = 3$,
4. any argument's term of α_i^j includes at most one empty list *nil*.

If we can regard all 2^n programs as hypotheses, then the program of $\mathcal{WR}_{\{d,c\}}$ is constructed from finite good examples $G = \{\alpha_1, \cdots, \alpha_k\}$ described by dotted pairs as follows: First, we obtain $G^j = \{\alpha_1^j, \cdots, \alpha_k^j\}$ from G for any j $(1 \leq j \leq 2^n)$. Then, by modifying $FIN\_CST\_\mathcal{WR}_{\{d\}}$ as Figure 2, we can construct P^j from $lgg(G^j)$ for any j. Note that an intended program is included in $\{P^j \mid 1 \leq j \leq 2^n\}$. Furthermore, the finiteness of the number of good examples are guaranteed by the following two corollaries of Lemma 6 and Theorem 2.

Corollary 1 Let Σ be an alphabet $\{a_1, \cdots, a_m, nil\}$. For $\mathcal{WR}_{\{d,c\}}$, the number of ground atoms under Σ such that $|t_i| \leq 5$ and t_i includes at most one empty list *nil* for any i $(1 \leq i \leq n)$ is at most $(1 + 2m^2 + 2m^3 + 2m^4)^n$.

Corollary 2 Let $\Sigma = \{a_1, a_2, a_3, a_4, nil\}$. Suppose that $P \in \mathcal{WR}_{\{d\}}$ and $\{\beta\} = U_P{\uparrow}3 - U_P{\uparrow}2$. Also suppose that $G = \{\alpha_1, \cdots, \alpha_k\}$ is the set of all good examples $p(t_1, \cdots, t_n)$ for P described by only dotted pairs under Σ such that $|t_i| \leq 5$ for any i $(1 \leq i \leq n)$. Then, there exists an index l $(1 \leq l \leq 2^n)$ such that $lgg(G^j) \equiv \beta$, where $G^j = \{\alpha_1^j, \cdots, \alpha_k^j\}$.

However, it is not efficient to consider all 2^n programs as hypotheses. Then, we design the algorithm $FIN\_CST\_\mathcal{WR}_{\{d,c\}}$ as Figure 4. Note that the algorithm $PROPOSE\_\mathcal{WR}_{\{d,c\}}$ as Figure 3 is an extended algorithm of $PROPOSE\_\mathcal{WR}_{\{d\}}$ as Figure 1. On the other hand, the algorithm *recursion_check* and *variable_check* is applied in order to select a program in the family of 2^n programs.

For the algorithm *recursion_check*, the following lemma holds:

Lemma 7 Let $lgg(G^i)$ in Figure 4 be an atom $p(v_1, \cdots, v_n)$. Then, the following statements are equivalent:

1. $lgg(G^i) \equiv \beta^i$.
2. For any l such that $v_l = d(W_l, v_l')$ and v_l' includes no variable W_l, if the variable W_l is included in v_m $(1 \leq m \leq n)$, then W_l is the leftmost variable in v_m.

Proof Suppose that there exist indexes l and m $(1 \leq l, m \leq n)$ such that

1. $v_l = d(W_{l1}, d(W_{l2}, X_l)), v_m = d(W_{m1}, d(W_{l1}, X_m))$,
2. $v_l = c(W_{l1}, c(W_{l2}, X_l)), v_m = d(W_{m1}, d(W_{l1}, X_m))$,

```
Algorithm $PROPOSE\_\mathcal{WR}_{\{d,c\}}(\beta, \{C_1, C_2\})$
input $\beta = p(t_1, \cdots, t_n)$ : atom
output $P = \{C_1, C_2\} \in \mathcal{WR}_{\{d,c\}}$
for $i = 1$ to $n$
    if $t_i$ is of the form $d(W_1^i, \cdots)$ then
        $head\_arg_i := d(W_1^i, X_i)$;            /* $X_i$ is a new variable */
    else if $t_i$ is of the form $c(W_1^i, \cdots)$ then
        $head\_arg_i := c(W_1^i, X_i)$;
    else
        $head\_arg_i := X_i$;
    end if
end for
$head := p(head\_arg_1, \cdots, head\_arg_n)$;
$body := p(X_1, \cdots, X_n)$;
$C_1 := head \leftarrow body$;
$tmp\_C_2 := (body)\theta$, where $\beta = (head)\theta$;
$C_2 := (body)\sigma$, where $tmp\_C_2 = (head)\sigma$;
output $P = \{C_1, C_2\}$
```

Figure 3: Algorithm $PROPOSE\_\mathcal{WR}_{\{d,c\}}$

3. $v_l = d(W_{l1}, d(W_{l2}, X_l)), v_m = c(W_{m1}, c(W_{l1}, X_m))$,
4. $v_l = c(W_{l1}, c(W_{l2}, X_l)), v_m = c(W_{m1}, c(W_{l1}, X_m))$,

where $W_{l1} \neq W_{l2}$, $W_{l1} \neq W_{m1}$, and $W_{l2} \neq W_{m1}$.

For the case 1, the following program is constructed by $PROPOSE\_\mathcal{WR}_{\{d,c\}}$:

$$P^i = \left\{ \begin{array}{l} p(t_1, \cdots, t_n) \leftarrow p(X_1, \cdots, X_n) \\ p(s_1, \cdots, s_n) \end{array} \right\}.$$

For an index l, $t_l = d(W_{l1}, X_l)$ and $s_l = X$. For an index Y, $t_m = d(W_{m1}, X_m)$ and $s_m = Y$. Then, $u_l = d(W_{l1}, d(W'_{l1}, X))$ and $u_m = d(W_{m1}, d(W'_{m1}, Y))$. Hence, $lgg(G^i) \not\equiv \beta^i$. For the cases 2 ,3 and 4, we can also show that $lgg(G^i) \not\equiv \beta^i$.

As contraposition of the above statement, if $lgg(G^i) \equiv \beta^i$, then there exist no indexes l and m such that v_l and v_m satisfy one of the above cases 1, 2, 3 and 4. In other words, if $lgg(G^i) \equiv \beta^i$, then, for any l $(1 \leq l \leq n)$ such that $v_l = d(W_{l1}, v'_l)$ and v'_l includes no variable W_{l1}, there exists no index m $(1 \leq m \leq n)$ such that $v_m = d(W_{m1}, v'_m)$ and v'_m includes no variable W_{l1}. For such v_l, if the variable W_{l1} is included in v_m then W_l is the leftmost variable in v_m. Hence, if $lgg(G^i) \equiv \beta^i$ and the variable W_l is included in v_m then W_l is the leftmost variable in v_m.

On the other hand, by the definition of good examples and by the supposition such that G^i is the set of all good examples under $\Sigma = \{a_1, a_2, a_3, a_4, nil\}$, if $v_l = d(W_{l1}, d(W_{l2}, X))$ and $v_m = d(W_{l1}, d(W_{m2}, Y))$, then $W_{l2} = W_{m2}$. Hence, the converse also holds. □

In other words, the statement $lgg(G^i) \not\equiv \beta^i$ means that P^i is too general as a program which explains the set G of good examples.

In the algorithm *variable_check*, $\#var(\alpha)$ means the number of variables appearing in an atom α. For the algorithm *variable_check*, the following lemma holds:

Lemma 8 For indexes i and j $(1 \leq i, j \leq 2^n)$, suppose that $lgg(G^i) \equiv \beta^i$ and $lgg(G^j) \equiv \beta^j$. If $\#var(lgg(G^i)) \leq \#var(lgg(G^j))$, then $dot(M_{P^i}) \subseteq dot(M_{P^j})$.

```
**Algorithm** $FIN\_CST\_\mathcal{WR}_{\{d,c\}}(G, P)$
**input** $G = \{\alpha_1, \cdots, \alpha_k\}$, where $\alpha_l = p(t^l_1, \cdots, t^l_n)$ $(1 \leq l \leq k)$ :
        set of all good examples for $P$ described by dotted pairs
        under $\{a_1, a_2, a_3, a_4, nil\}$ such that $|t_i| \leq 5$ for any $i$ $(1 \leq i \leq n)$
**output** $P \in \mathcal{WR}_{\{d,c\}}$
**read** $G$;
$recursion\_check(G, I)$;
$variable\_check(G, I, J)$;
**select** $j \in J$;
$G^j := \{\alpha^j_1, \cdots, \alpha^j_k\}$;
$PROPOSE\_\mathcal{WR}_{\{d,c\}}(lgg(G^j), P^j)$;
**output** $P^j$

**Algorithm** $recursion\_check(G, I)$
**input** $G = \{\alpha_1, \cdots, \alpha_k\}$ : set of atoms
**output** $I$ : set of indexes
$I := \phi$;
**for** $i = 1$ **to** $2^n$
    $G^i := \{\alpha^i_1, \cdots, \alpha^i_k\}$;
    $PROPOSE\_\mathcal{WR}_{\{d,c\}}(lgg(G^i), P^i)$;
    $\{\beta^i\} := U_{P^i} \uparrow 3 - U_{P^i} \uparrow 2$;
    **if** $lgg(G^i) \equiv \beta^i$ **then**
        $I := I \cup \{i\}$;
    **end if**
**end for**

**Algorithm** $variable\_check(G, I, J)$
**input** $G = \{\alpha_1, \cdots, \alpha_k\}$ : set of atoms
        $I$ : set of indexes
**output** $J$ : set of indexes
$J := I$;
**while** $I = \phi$ **do**
    $H := I$;
    **select** $i \in I$;
    $G^i := \{\alpha^i_1, \cdots, \alpha^i_k\}$;
    **while** $H = \phi$ **do**
        **select** $h \in H$;
        $G^h := \{\alpha^h_1, \cdots, \alpha^h_k\}$;
        **if** $\#var(lgg(G^h)) < \#var(lgg(G^i))$ **then**
            $J := J - \{i\}$;
        **end if**
        $H = H - \{h\}$;
    **end while**
    $I = I - \{i\}$;
**end while**
```

Figure 4: Algorithm $FIN\_CST\_\mathcal{WR}_{\{d,c\}}$

Proof Let $lgg(G^i)$ and $lgg(G^j)$ be of the forms $p(v_1^i, \cdots, v_n^i)$ and $p(v_1^j, \cdots, v_n^j)$. Suppose that $\alpha \in dot(M_{P^i})$.

If $\#var(lgg(G^i)) \leq \#var(lgg(G^j))$, then there exist indexes l and m $(1 \leq l, m \leq n)$ such that, for $e_1, e_2, f_1, f_2 \in \{d, c\}$, v_l^i, v_m^i, v_l^j, and v_m^j satisfy either of the following conditions:

1. $v_l^i = e_1(W_{l1}, e_1(W_{l2}, X_l))$, $v_m^i = e_2(W_{l1}, e_2(W_{l2}, X_m))$, $v_l^j = f_1(W_{l1}, f_1(W_{l2}, X_l))$, and $v_m^j = f_2(W_{m1}, f_2(W_{m2}, X_m))$, where $W_{l1} \neq W_{m1}$.
2. $v_l^i = e_1(W_{l1}, e_1(W_{l2}, X_l))$, $v_m^i = e_2(W_{l1}, e_2(W_{l2}, X_l))$, $v_l^j = f_1(W_{l1}, f_1(W_{l2}, X_l))$, and $v_m^j = f_2(W_{l1}, f_2(W_{l2}, X_m))$, where $X_l \neq X_m$.

For the case 1, since $lgg(G^i) \equiv \beta^i$ and $lgg(G^j) \equiv \beta^j$, $W_{l1}, W_{l2}, W_{m1}, W_{m2}$ are mutually distinct. By the supposition, $\alpha^i \in M_{P^i}$. Then, by the construction P^j, $\alpha^j \in M_{P^j}$. Hence, $\alpha \in dot(M_{P^j})$.

For the case 2, we have the same proof. □

Hence, if $lgg(G^i) \equiv \beta^i$, $lgg(G^j) \equiv \beta^j$, and $\#var(lgg(G^i)) = \#var(lgg(G^j))$, then $dot(M_{P^i}) = dot(M_{P^j})$.

By the above lemma, if $|J| > 1$ for *recursion_chech*(G, I) and *variable_check*(G, I, J), then, for any $j \in J$, P^j is regarded as an intended program of theory-generating abduction for $\mathcal{WR}_{\{d\}}$.

Hence, the following theorem holds:

Theorem 3 Let $P \in \mathcal{WR}_{\{d,c\}}$. Then, by using the algorithm *FIN_CST_*$\mathcal{WR}_{\{d,c\}}(G, P)$ of theory-generating abduction for $\mathcal{WR}_{\{d,c\}}$ as Figure 4, P is constructed correctly from finite good examples G described by dotted pairs.

Example 5 Let G be the following set of all good examples for P_5 under $\{1, 2, 3, 4, nil\}$ described by dotted pairs:

$$G = \left\{ \begin{array}{l} p(d(1, d(2, nil)), d(1, d(2, d(3, nil))), d(2, d(1, nil))) \\ p(d(2, d(3, nil)), d(2, d(3, nil)), d(3, d(2, nil))) \\ p(d(4, d(1, nil)), d(4, d(1, d(2, d(3, nil)))), d(1, d(4, nil))) \\ \cdots\cdots\cdots\cdots\cdots\cdots \end{array} \right\}.$$

Then, we obtain the following $lgg(G^j)$ $(1 \leq j \leq 8)$:

$$\begin{array}{l} lgg(G^1) = p(d(W_1, d(W_2, nil)), d(W_1, d(W_2, X)), d(W_2, d(W_1, nil))), \\ lgg(G^2) = p(c(W_2, c(W_1, nil)), d(W_1, d(W_2, X)), d(W_2, d(W_1, nil))), \\ lgg(G^3) = p(d(W_1, d(W_2, nil)), c(W_3, c(W_4, X)), d(W_2, d(W_1, nil))), \\ lgg(G^4) = p(d(W_1, d(W_2, nil)), d(W_1, d(W_2, X)), c(W_1, c(W_2, nil))), \\ lgg(G^5) = p(c(W_2, c(W_1, nil)), c(W_3, c(W_4, X)), d(W_2, d(W_1, nil))), \\ lgg(G^6) = p(c(W_2, c(W_1, nil)), d(W_1, d(W_2, X)), c(W_1, c(W_2, nil))), \\ lgg(G^7) = p(d(W_1, d(W_2, nil)), c(W_3, c(W_4, X)), c(W_1, c(W_2, nil))), \\ lgg(G^8) = p(c(W_2, c(W_1, nil)), c(W_3, c(W_4, X)), c(W_1, c(W_2, nil))). \end{array}$$

By *recursion_check*(G, I), we obtain the set $I = \{4, 5, 7\}$ of indexes. By *variable_check*(G, I, J), we also obtain the set $J = \{4\}$ of indexes. Hence, we obtain the following program P_5^4 by the algorithm *FIN_CST_*$\mathcal{WR}_{\{d,c\}}$:

$$P_5^4 = \left\{ \begin{array}{l} p(d(W_1, X_1), d(W_1, X_2), c(W_1, X_3)) \leftarrow p(X_1, X_2, X_3) \\ p(nil, X, nil) \end{array} \right\}.$$

The program P_5^4 means that the first argument's term is the prefix of the second argument's term, and it is also the reversal of the third argument's term.

7 Conclusion

We have introduced a subclass of logic programs, called *weakly reducing programs* $\mathcal{WR}_{\{d\}}$ *with dotted pairs*, and formulated the concept of good examples for $\mathcal{WR}_{\{d\}}$. Then, we have designed the algorithm of theory-generating abduction for $\mathcal{WR}_{\{d\}}$, and shown that the program of the class $\mathcal{WR}_{\{d\}}$ is constructed correctly from finite good examples. Furthermore, we have extended the class $\mathcal{WR}_{\{d\}}$ to *weakly reducing* programs $\mathcal{WR}_{\{d,c\}}$ *with dotted pairs and concatenations* by introducing a function *concatenation*, and investigated theory generating abduction for $\mathcal{WR}_{\{d,c\}}$. For the class $\mathcal{WR}_{\{d,c\}}$, we have formulated the concept of *good examples described by dotted pairs*, and designed the algorithm of theory-generating abduction. Then, we have shown that the program of the class $\mathcal{WR}_{\{d\}}$ is constructed correctly by this algorithm from finite good examples. We have also shown that this algorithm also determines which of arguments' terms are described by concatenations.

The class given in this paper is based on logic programs for list processing. Introducing functions will be applied to other classes of logic programs. Also the class given in this paper is based on single recursion of logic programs. It is a future work to extend the class and to improve algorithms in order to apply to other data structures and recursions. Furthermore, in this paper, we have assumed that only good examples are given in theory-generating abduction. It is also an important future work how to select good examples from all examples.

References

[ALLM94] Aha, D. W., Lapointe, S., Ling, X. C. and Matwin, S.: *Inverting implication with small training sets*, Proceedings of European Conference on Machine Learning (1994), Lecture Notes in Artificial Intelligence **784**, 31–43, 1994.

[ASOI94] Arimura, H., Shinohara, S., Otsuki, S. and Ishizaka, I.: *A generalization of the least general generalization*, Machine Intelligence **13**, 1994.

[FKW93] Freivalds, R., Kinber, E. B. and Wiehagen, R.: *On the power of inductive inference from good examples*, Theoretical Computer Science **110**, 131–144, 1993.

[Hir93] Hirata, K.: *A classification of abduction: abduction for logic programming*, Machine Intelligence **14** (to appear).

[Hir94] Hirata, K.: *Rule-generating abduction for recursive Prolog*, Proceedings of the 4th International Workshop on Analogical and Inductive Inference, Lecture Notes in Artificial Intelligence **872**, 121–136, 1994.

[LNW94] Lange, S., Nessel, J. and Wiehagen, R.: *Language learning from good examples*, Proceedings of the 5th International Workshop on Algorithmic Learning Theory, Lecture Notes in Artificial Intelligence **872**, 423–437, 1994.

[Lin91] Ling, C. X.: *Inductive learning from good examples*, Proceedings of the 12th International Joint Conference on Artificial Intelligence, 751–756, 1991; revised version in [Mug92].

[Llo87] Lloyd, J. W.: *Foundations of logic programming* (second extended edition), Springer-Verlag, 1987.

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

[Muk92] Mukouchi, Y.: *Characterization of finite identification*, Proceedings of the 3rd International Workshop on Analogical and Inductive Inference, Lecture Notes in Artificial Intelligence **642**, 260 – 267, 1992.

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

[Plo70] Plotkin, G. D.: *A note on inductive generalization*, Machine Intelligence **5**, 153–163, 1970.

[Shi90] Shinohara, T.: *Inductive inference of monotonic formal systems from positive data*, Proceedings of the 1st International Workshop on Algorithmic Learning Theory, 339–351, 1990.