Sufficiency of Operators Identification and Inter-construction in Inverting Resolution

Zeng, chao
Department of Information Systems, Kyushu University

Arikawa, Setsuo
Research Institute of Fundamental Information Science Kyushu University

https://doi.org/10.5109/3131

出版情報：Bulletin of informatics and cybernetics. 24 (3/4), pp.111-120, 1990-04-03. Research Association of Statistical Sciences
バージョン：
権利関係：

RIFIS-TR-CS-24

RIFIS Technical Report

Sufficiency of Operators Identification and Inter-construction in Inverting Resolution

Chao Zeng
Setsuo Arikawa

April 3, 1990

Research Institute of Fundamental Information Science
Kyushu University 33
Fukuoka 812, Japan
E-mail: ayumi@rifis.sci.kyushu-u.ac.jp Phone: 092(641)1101 Ext.4471

Sufficiency of Operators Identification and Inter-construction in Inverting Resolution

Chao Zeng and **Setsuo Arikawa**
Department of Information Systems
Kyushu University 39, Kasuga 816

March, 1990

Abstract

In this paper we give a formal definition of examples(or training instances) that are widely used in the field of concept learning. Based on this definition, we discuss the operators introduced by Muggleton[2] in Duce system, especially the sufficiency of operators identification and inter-construction in the sense that for any given ground definite program P we can give out a set of examples(named characteristic sample) from which we can learn P using these two operators. For this purpose, we define the concept of unique definition for identification and inter-construction which corresponds to the isolated reference defined by Muggleton[4] for absorption and intra-construction. We also discuss the independence of Duce operators.

1 Introduction

Muggleton[2] has proposed a machine learning system Duce based on a propositional logic. Duce can enrich the learner's vocabulary by suggesting new descriptive terms(new intermediate concepts) to the user. For a practical use of such an inference system it is important to evaluate the inference method used in the system[6]. We consider the sufficiency of Duce operators as an evaluation criterion. Some operators uesd in Duce have been proved sufficient for a class of logic programs in the other framework[4]. In this paper, we prove that operators identification and inter-construction are sufficient for learning ground definite programs from examples we are defining. We start with recalling the operators in Muggleton[2] by examples. All the atoms in the examples below are ground.

1. Identification

$$\begin{array}{lll} P': & C & x \leftarrow a, b, c, d, e \\ & C_1 & x \leftarrow a, b, p \\ & & \Downarrow \\ P: & C_1 & x \leftarrow a, b, p \\ & C_2 & p \leftarrow c, d, e \end{array}$$

The operator identification(Ident, for short) transforms P' to $P = (P' - \{C\}) \cup \{C_2\}$, and $Ident^{-1}$ is called the inverse of Ident that transforms P to $P' = (P - \{C_2\}) \cup \{C\}$. There is a question whether there are other rules with head p in the rule base. If there are not, the new rule C_2 is valid. Otherwise it will be verified by the oracle. This is the same as that pointed out in [2] about absorption.

2. **Inter-construction**

$$
\begin{array}{llll}
P': & BB: & B_1 & x \leftarrow b, c, d, e \\
 & & B_2 & y \leftarrow a, b, d, f \\
 & & & \Downarrow \\
P: & CC: & C_1 & x \leftarrow c, e, p? \\
 & & C_2 & y \leftarrow a, f, p? \\
 & A: & & p? \leftarrow b, d
\end{array}
$$

Note here that p? indicates that p is a new predicate symbol, i.e., a new intermediate concept, produced by the operator. The operator inter-construction(Inter, for short) transforms P' to $P = (P' - BB) \cup CC \cup \{A\}$ and $Inter^{-1}$, the inverse of Inter, transforms P to $P'=(P - (CC \cup \{A\})) \cup BB$.

3. **Intra-construction**

$$
\begin{array}{llll}
P': & BB: & B_1 & x \leftarrow b, c, d, e \\
 & & B_2 & x \leftarrow a, b, d, f \\
 & & & \Downarrow \\
P: & CC: & C_1 & p? \leftarrow c, e \\
 & & C_2 & p? \leftarrow a, f \\
 & A: & & x \leftarrow b, d, p?
\end{array}
$$

The operator intra-construction(Intra, for short) transforms P' to $P = (P' - BB) \cup CC \cup \{A\}$ and $Intra^{-1}$, the inverse of Intra, transforms P to $P' = (P - (CC \cup \{A\})) \cup BB$.

4. **Absorption**

$$
\begin{array}{llll}
P': & C & & x \leftarrow a, b, c, d, e \\
 & C_1 & & p \leftarrow a, b, c \\
 & & & \Downarrow \\
P: & C_1 & & p \leftarrow a, b, c \\
 & C_2 & & x \leftarrow d, e, p
\end{array}
$$

The operator absorption(Abs, for short) transforms P' to $P = (P' - \{C\}) \cup \{C_2\}$ and Abs^{-1}, the inverse of Abs, transforms P to $P' = (P - \{C_2\}) \cup \{C\}$.

5. **Truncation**

$$
\begin{array}{lll}
P': & C_1 & x \leftarrow a, b, c, d \\
 & C_2 & x \leftarrow a, c, e, f \\
 & & \Downarrow \\
P: & C & x \leftarrow a, c
\end{array}
$$

Note that when we apply truncation, we must check whether the resultant rule C conflicts with any other rule in the rule base.

6. **Dichotomisation**

$$
\begin{array}{lll}
P': & C_1' & x \leftarrow a, b, c, d \\
 & C_2' & \neg x \leftarrow a, c, e, f \\
 & C_3' & \neg x \leftarrow a, b, c, g \\
 & & \Downarrow \\
P: & C_1 & x \leftarrow a, c, p? \\
 & C_2 & \neg x \leftarrow a, c, \neg p? \\
 & C_3 & p? \leftarrow b, d \\
 & C_4 & \neg p? \leftarrow e, f \\
 & C_5 & \neg p? \leftarrow b, g
\end{array}
$$

The operator dichotomisation transforms P' to P.

2 Sufficiency of Identificatin and Inter-construction

In this section, we define the concept of training instance set as the set of examples given by user to learn the desired concept. We prove that we can learn any ground definite program P from a particular training instance set, which is called characteristic sample of P, using identification and inter-construction. Here we use the terminology in [1].

Definition 1 A *ground atom* is an atom not containing variables.

Definition 2 A *definite clause* is a clause of the form

$$h \leftarrow b_1, b_2, \cdots, b_n$$

which contains precisely one atom (h) in its consequent. h is called the *head* and $b_1, b_2, \cdots, b_n$ is called the *body* of the clause. A *ground definite clause* is a definite clause that all atoms in it are ground.

Definition 3 A *definite program* is a finite set of definite clauses. A *ground definite program* is a finite set of ground definite clauses.

Definition 4 We say that definite clause

$$p \leftarrow \alpha \in P$$

defines the predicate symbol p in P, where α is a conjunction of atoms. Note here that α may be empty.

Definition 5 Let E be a ground definite program.
An *inverse derivation of E based on identificatin and inter-construction* is a mixed sequence of transformations by these two operators

$$E \rightarrow P_1 \rightarrow \cdots \rightarrow P_n$$

from E into other definite program P_n, and the P_n is called an *inverse derivative of E.*

Definition 6 Let E be a ground definite program. We say that a definite program *P is learnable from E by identification and inter-construction* if it is an inverse derivative of E.

Of course, from a ground definite program many different inverse derivatives may be produced by different search strategies or different orders of operator applications. Here we do not consider them but put stress on the existence of the desired inverse derivative.

Definition 7 Let P be a definite program and a predicate symbol p be defined by $p \leftarrow \alpha$ in P. We say that p is an *intermediate (predicate) symbol* in P if p occurs in the *body* of some other clauses in P.

In order to show the sufficiency of identification and inter-construction we need to change the definition of the example set because system based on just these two operators can not learn any clause with nonempty body from the unit clauses. We now give a new definition of examples, which should be more natural and usable than the original one in [3].

Definition 8 Let E be a ground definite program. We say that E is a *training instance set* if E contains no intermediate symbols.

Example 1 Consider the following ground definite programs:

E: $p \leftarrow a, b, c, d$; $q \leftarrow a, c, e, f$; $p \leftarrow c, g, h$; $q \leftarrow e, f, l$

E': $p \leftarrow a, b, q$; $q \leftarrow a, c, e, f$; $p \leftarrow c, g, h$

E contains no intermediate symbols in it. Hence by the definition, E is a training instance set. On the other hand, E' contains an intermediate symbol q in it. Hence it is not a training instance set.

Definition 9 For a ground definite program E, we define

$$Duce_{(Ident,Inter)}(\mathrm{E}) = \{\mathrm{P} \mid \mathrm{P} \text{ is an inverse derivative of E}\},$$

and call it an *hypothesis space of E.*

The $Duce_{(Ident,Inter)}(E)$ is an algorithm that produces all the inverse derivatives of the input E, and it contains all the definite programs learnable from E. We are now in a position to define those ground definite programs from which the desired definite program is learnable. Clearly, for any definite program, such ground definite programs are not unique.

Definition 10 Given a definite program P, we say that E is a *characteristic sample of P* for algorithm $Duce_{(Ident,Inter)}$ if E is a *training instance set* and $P \in Duce_{(Ident,Inter)}(E)$.

By the above definitions, now the sufficiency of identificatin and inter-construction becomes a question whether we can give out a characteristic sample E for any definite program P. The answer will be proved yes.

Definition 11 If $p \leftarrow \alpha$ is only the clause in definite program P that *defines* the predicate symbol p then we say that P contains a *unique definition* of p.

As in Example 1, E' contains a unique definition of the predicate symbol q, but does not E.

Remark 1 We use *vocab(P)* to indicate the set of all predicate symbols in P.

Remark 2 Let C_1 be a clause which contains p in its body and C_2 be a clause which defines p. If $Ident^{-1}$ is applied to $P \supseteq \{C_1, C_2\}$ to produce $P' = (P - \{C_2\}) \bigcup \{C_1 * C_2\}$, then the operator $Ident^{-1}$ reduces the number of clauses which define predicate symbol $p \in vocab(P)$ by one. Note here that the number of clauses in P' is equal to that in P.

Remark 3 Let A=p$\leftarrow \alpha$ be the clause that uniquely defines p in P and CC be the set of all clauses in P which contain p in their bodies. If $Inter^{-1}$ is applied to $P \supseteq (\{A\} \bigcup CC)$ to produce $P' = (P - (\{A\} \bigcup CC)) \bigcup BB$, then P' will never contain the predicate symbol p. Note also that the number of clauses in P' is one less than that in P.

Now we can use the following algorithm $Char_{(Ident,Inter)}$ to generate a characteristic sample of a given ground definite program P.

Algorithm Char$_{(Ident,Inter)}$(P)
begin $Char_{(Ident,Inter)}(P)$
  let i=0,P_0=P
  until P_i is a training instance set **do**
    if $\exists A \in P_i$ such that A is a unique definition of p in P_i
    then
      P_{i+1} is the result of applying $Inter^{-1}$ to remove p in P_i
    else
      P_{i+1} is the result of applying $Ident^{-1}$ to remove the definition A from P_i
    let i=i+1
  done
  E is P_i
  return(E)
end $Char_{(Ident,Inter)}(P)$

We will prove that the output E is a characteristic sample of input P.

Example 2 Consider the following input definite program P:

P: $p \leftarrow a, b$
$a \leftarrow c, d, e, f$
$c \leftarrow r, s, t$
$c \leftarrow s, h$

The *algorithm* $Char_{(Ident,Inter)}(P)$ works as follows:

P_0=P: $p \leftarrow a, b$
$a \leftarrow c, d, e, f$
$c \leftarrow r, s, t$
$c \leftarrow s, h$

At the first step, P_0(i.e.,P) is not a training instance set. Hence we choose $a \leftarrow c, d, e, f$ as A which is a unique definition of a. Then the algorithm uses $Inter^{-1}$ to remove predicate symbol a in P_0 and produces

P_1: $p \leftarrow c, d, e, b$
$c \leftarrow r, s, t$
$c \leftarrow s, h$.

P_1 is not a training instance set and clause $c \leftarrow r, s, t$ is not a unique definition of predicate symbol c. Hence we choose $c \leftarrow r, s, t$ as C_2 and $p \leftarrow c, d, e, b$ as C_1 which contains c in its body. Then the algorithm uses $Ident^{-1}$ to remove $c \leftarrow r, s, t$ from P_1 and produces

P_2: $p \leftarrow c, d, e, b$
$p \leftarrow r, s, t, d, e, b$
$c \leftarrow s, h$.

P_2 is not a training instance set and now $c \leftarrow s, h$ is a unique definition of predicate symbol c. Hence we choose $c \leftarrow s, h$ as A which is a unique definition of a. Then the algorithm uses $Inter^{-1}$ to remove predicate symbol c in P_2 to get

P_3: $p \leftarrow s, h, c, d, e, b$
$p \leftarrow r, s, t, d, e, b$,

which is a training instance set. Then the *algorithm* $Char_{(Ident,Inter)}(P)$ terminates and returns P_3 as the output E.

Essentially, the characteristic sample that the algorithm $Char_{(Ident,Inter)}$ looks for is a set of examples which contains only the clauses that are described by the low level features(bodies) and the high level concepts(heads), but not by any intermediate concept(i.e., those predicate symbols appears in the body of some clauses and are defined by some other clauses in the same program)[5]. Now we can use this algorithm to prove that there is a characteristic sample for any ground definite program, and to generate it.

In the following, we apply the operators identification and inter-construction inversely to derive and create all the clauses of the definite program from E. Of course, the search method for the inverse derivation process is non-deterministic, to which we do not refer any more in this paper.

Theorem 1 For any ground definite program P, the algorithm $Char_{(Ident,Inter)}(P)$ terminates, and E=$Char_{(Ident,Inter)}(P)$ is a characteristic sample of P.

Proof First we prove that the algorithm $Char_{(Ident,Inter)}(P)$ terminates in a finite time. Suppose p is an intermediate predicate symbol in P. Then the definition of p in P contains at least one clause. Hence there are two cases to be considered:

1. If the definition A is a unique definition of p, then at the stage of **then statement**, we can erase the occurrence of p.

2. If there are several definitions, then they will be removed one by one by the **else statement**, and the last one will be removed by the above 1.

The processes 1 and 2 above may continue until P_i becomes a training instance set. Then the algorithm $Char_{(Ident,Inter)}(P)$ terminates. Because P is a finite set of ground definite clauses, the algorithm will terminate in a finite time.

Secondly we prove that E=$Char_{(Ident,Inter)}(P)$ is a characteristic sample of P. Since a sequence of programs

$$P \rightarrow P_1 \rightarrow \cdots \rightarrow P_{n-1} \rightarrow E = P_n$$

from P to E is obtained, and $Inter^{-1}$ or $Ident^{-1}$ is applied in each step. Hence we have $P \in Duce_{(Ident,Inter)}(E)$. By the termination condition, E is a training instance set. Then by Definition 10, E=$Char_{(Ident,Inter)}(P)$ is a characteristic sample of P. **Q.E.D**

Theorem 2 Let P be a ground definite program, $E = Char_{(Ident,Inter)}(P)$ and P_i be the set of intermediate symbols in P. Then

$$\mid E \mid = \mid P \mid - \mid P_i \mid,$$

where $\mid X \mid$is the number of clauses in X and $\mid E \mid$ is called the size of the characteristic sample E.

Proof According to the Remark 2, $Ident^{-1}$ is applied to $P \supseteq \{C_1, C_2\}$ to produce

$$P' = (P - \{C_2\}) \cup \{C_1 * C_2\},$$

but the number of clauses in P does not decrease. Hence $\mid P' \mid = \mid P \mid$.

According to the Remark 3, $Inter^{-1}$ is applied to $P \supseteq (\{A\} \cup CC)$ to produce

$$P' = (P - (\{A\} \cup CC)) \cup BB,$$

and the number of clauses in P decreases by one. Hence $\mid BB \mid = \mid CC \mid$, and $\mid P' \mid = \mid P \mid -1$.

In the proof of Theorem 1, we have shown that the unique definition of an intermediate predicate symbol in P is removed using $Inter^{-1}$, and others are obtained by applying $Ident^{-1}$. If there are $\mid P_i \mid$ intermediate predicate symbols in P, then, until the algorithm terminates, $Inter^{-1}$ must be applied $\mid P_i \mid$ times. Hence $\mid E \mid = \mid P \mid - \mid P_i \mid$. **Q.E.D**

Example 3 As in Example 2, the algorithm applies $Inter^{-1}$ twice and hence $\mid P \mid$=4 and $\mid P_i \mid$=2. The returned value E has two clauses. Thus $\mid E \mid = \mid P_3 \mid = 2$, and $\mid E \mid = \mid P \mid - \mid P_i \mid = 2$.

From Theorem 1 and 2, we have seen that the size of example set for learning a definite program is bounded by the size of the desired program. So when the given example set is large enough, it may contain at least one characteristic sample of the desired definite program. Thus it is theoretically guaranteed that by using the operators identification and inter-construction we can learn the desired definite program.

3 Independence of Duce Operators

We have proved that operators identification and inter-construction also have the sufficiency. For Duce system, Muggleton[2] gave out six operators in all. In this section, we discuss the independence of some of them. We show that the operator truncation can be replaced by inter-construction or intra-construction if we introduce some minute techniques to the original system.

3.1 Replacing Truncation by Inter-construction

Since we restrict our discussion in propositional logic, we do not need to consider the recursion. Hence we can assume the following condition in Duce system.

Condition: We suppose there is no recursion, that is, there is no such rule whose head occurs in the body as a predicate symbol in the rule base. If there are such clauses, we remove them from rule base.

Under this condition, we can call the oracle(user) to give the new concept an appropriate name in the process of executing inter-construction to realize truncation as shown in the following example.

Example 4 Consider the following example in Figure 1, where we want to learn $P : x \leftarrow a, c$ from

$$\begin{array}{ll} P': & x \leftarrow a, b, c, d \\ & x \leftarrow a, c, j, k \end{array}$$

by using truncation, or using inter-construction instead of truncation.

$$\begin{array}{lll} P': & C & x \leftarrow a, b, c, d \\ & C_1 & x \leftarrow a, c, j, k \end{array} \quad \underset{\text{(truncation)}}{\Longrightarrow} \quad P: \; x \leftarrow a, c$$

$$\Downarrow \text{ (inter-construction)}$$

$$\begin{array}{lll} P: & C_1 & x \leftarrow b, d, z? \\ & C_2 & x \leftarrow j, k, z? \\ & A & z? \leftarrow a, c \end{array} \quad \underset{\text{(condition)}}{\overset{z? \leftarrow x}{\Longrightarrow}} \quad P: \; x \leftarrow a, c$$

Figure 1

As shown in the figure, we can get $P : x \leftarrow a, c$ by an application of truncation. But we can also get the same result in two steps, that is, at the first step inter-construction is applied. Since inter-construction generates new concept, and at the second step the new concept is given the name x depending on the oracle(user). Then, according to the condition, the recursive clauses in the rule base are removed, and P: $x \leftarrow a, c$ is obtained. Note that, in general, the operator inter-construction generates new concept, but when it is used as in the above and it generates no new concept, it works just like truncation.

3.2 Replacing Truncation by Intra-construction

In order to replace truncation by intra-construction, we need to introduce to the system a special predicate symbol T with the following condition.

Condition: When the T appears in the head of a clause, we remove the clause from the rule base, and when T appears in the body, we remove T from it.

Under this condition, we can call the oracle(user) to give the new concept the name T appropriately in the process of executing intra-construction to realize truncation as shown in the following example.

Example 5 Consider the following example in Figure 2, where we want to learn $P : x \leftarrow a, c$ from

$$\begin{array}{ll} P': & x \leftarrow a, b, c, d \\ & x \leftarrow a, c, j, k \end{array}$$

by using truncation, or using intra-construction instead of truncation.

$$\begin{array}{lll} P': & C & x \leftarrow a, b, c, d \\ & C_1 & x \leftarrow a, c, j, k \end{array} \quad \underset{\text{(truncation)}}{\Longrightarrow} \quad P: \ x \leftarrow a, c$$

$$\Downarrow \ \text{(intra-construction)}$$

$$\begin{array}{lll} P: & C_1 & z? \leftarrow b, d \\ & C_2 & z? \leftarrow j, k \\ & \text{A} & x \leftarrow a, c, z? \end{array} \quad \underset{\text{(condition)}}{\overset{z? \leftarrow \text{T}}{\Longrightarrow}} \quad P: \ x \leftarrow a, c$$

Figure 2

As shown in the figure, we can get $P : x \leftarrow a, c$ by an application of truncation. But we can also get the same result in two steps, that is, at the first step intra-construction is applied. Since intra-construction generates new concept, and at the second step the new concept is given the name T depending on the oracle(user). Then, according to the condition, the clauses with the special predicate symbol T in their heads and the occurrence of T in the body are removed, and P: $x \leftarrow a, c$ is obtained. Note also that, in general, the operator intra-construction generates new concept, but when it is used as in the above and it generates no new concept, it works just like truncation.

Thus the operator truncation is not necessary for Duce system, but it contributes to make the system more efficient. As known from the examples in [2], the tuncation operator reduces more symbols than the other operators.

4 Conclusion

In this paper, we have discussed the sufficiency of operators identification and inter-construction based on a new definition of training instance set. Although we have not discussed the operators absorption and intra-construction based on the new definition, a similar result may be obtained for them. As a future work, we are considering the same questions in the framework of first-order logic.

References

[1] J.W.Lloyd *(1987): Foundations of Logic Programming. Springer-Verlag,Germany (Second Version).*

[2] S.Muggleton *(1987): Duce,an oracle based approach to constructive induction. In IJCAI-87,Kaufmann,pp.287-292.*

[3] S.Muggleton and W.Buntine *(1988): Towards Constructive Induction in first-order predicate calculus. Turing Institute working paper.*

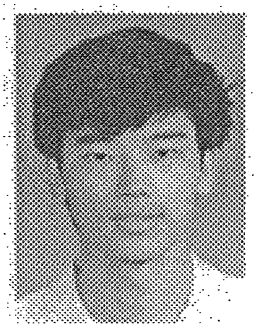
[4] S.Muggleton and W.Buntine *(1988): Machine Invention of first-order predicates by inverting resolution.In* Machine Learning 5 *,Kaufmann,pp.339-352.*

[5] L.M.Fu and B.G.Buchanan *(1985): Learning Intermediate Concepts in Constructing a Hierarchical Knowledge Base.in Proceedings of the Ninth International Joint Conference on Artificial Intelligence.Los Altos,CA:Kaufmann,(1985),pp.650-658.*

[6] D.Angluin and C.H.Smith *(1983): Inductive Inference: Theory and Methods. in ACM computing Surveys 15,237-269.*

[7] R.S.Michalski *(1983): A Theory and Methodology of Inductive Learning.in R.S.Michalski, J.G.Carbonell, and T.M.Mitchell, (Eds.),Machine Learning. Palo Alto:Tioga.*

About the Authors



Chao Zeng (曽　超) was born in Guizhou Province of China on February 14, 1965. He received the B.S. degree from Zhongshan University of China in 1985. Presently, he is a graduate student of Master Course in Information Systems, Kyushu University. His research interests are in machine learning and logic programming.

Setsuo Arikawa (有川節夫) was born in Kagoshima on April 29, 1941. He received the B.S. degree in 1964, the M.S. degree in 1966 and the Dr.Sci. degree in 1969 all in Mathematics from Kyushu University. Presently, he is Professor of Research Institute of Fundamental Inforamtion Science, Kyushu University. His research interests include algorithmic learning theory, logic and inference in AI, and information retrieval systems.

Research Institute of Fundamental Information Science, Kyushu University, Fukuoka 812, Japan.