INDUCTIVE INFERENCE OF FORMAL SYSTEMS FROM POSITIVE DATA

Shinohara, Takeshi
Computer Center, Kyushu University

https://doi.org/10.5109/13373

出版情報：Bulletin of informatics and cybernetics. 22 (1/2), pp.9-18, 1986-03. Research Association of Statistical Sciences
バージョン：
権利関係：

Bulletin of Informatics and Cybernetics Vol. 22, No. 1~2, 1986

INDUCTIVE INFERENCE OF FORMAL SYSTEMS FROM POSITIVE DATA

By

Takeshi SHINOHARA*

Abstract

A formal system, we deal with in this paper, is a set of formulas of the form $P_1(t_1) \leftarrow P_2(t_2)\ \&\ \cdots\ \&\ P_n(t_n)$, where $P_1, P_2, \cdots, P_n$ are predicate symbols and $t_1, t_2, \cdots, t_n$ are strings of constant symbols and variable symbols. The language defined by a formal system E is a set of constant strings t such that $P_i(t)$ is provable from E by using rules of modus ponens and substitutions of constant strings for variable symbols. We restrict formal systems so as to contain only two formulas of a predicate, and also restrict substitutions not to map any variable to empty string. The class of languages defined by our restricted formal systems is a natural extension of Angluin's pattern languages. We show that the class is inferable from positive data.

1. Introduction

Formal languages are mostly defined by using generation grammars and recognition automata. We know another kind of systems, elementary formal systems introduced by Smullyan [1], to define such formal languages. The concept of elementary formal systems is based on the following type of axiom schema to define a set S which occurs frequently in mathematical literature:

(1) a is an element of S,

(2) if x is an element of S, then so is $f(x)$, and

(3) only the elements that follow from (1) and (2) are in S.

Smullyan developed his recursive function theory by using the elementary formal systems with the rudimentary attributes. We can use the elementary formal systems to define formal languages.

Arikawa [2] studied languages defined by elementary formal systems and formal properties of restricted systems, called simple formal systems. The class of languages defined by simple formal systems is located exactly between the classes of context-free languages and context-sensitive languages.

In this paper, we introduce more restricted systems than Arikawa's simple formal systems, which will be called *primitive formal systems*, and show that the class of languages represented by them is inferable from positive data. Our restricted systems

* Computer Center, Kyushu University 91, Fukuoka 812, Japan.

have only one base step and one induction step. Angluin's pattern languages [3, 4], which the author has already studied from the viewpoint of practical applications of inductive inference [5, 6, 7, 8], can be regarded as languages represented by the simplest systems that have just one base step and have no induction step. Thus the class of languages defined by our systems is a natural extension of pattern languages.

2. Formal Systems and Languages

First, we introduce the notion of the elementary formal systems and define their languages.

DEFINITION 1. Let Σ be a finite set of symbols, and let X and D be countable sets of symbols, where Σ, X and D are mutually disjoint. Elements in Σ, X and D are called *constants*, *variables* and *predicates*, and denoted by $a, b, c, \cdots; x, y, z, x_1, x_2, \cdots$; and $P, Q, R, P_1, P_2, \cdots$, respectively.

According to Smullyan [1], positive integers called degrees are assigned to predicates. Here we assume that degrees of predicates are all one, that is, the numbers of arguments of our predicates are always 1.

DEFINITION 2. For a set S of symbols, we denote a set of all finite string over S byS^*. Strings in $(\Sigma \cup X)^*$ are called *terms*. When term t_1 is identical to t_2 except renaming of variables, we say t_1 is *equivalent* to t_2 and denote it by $t_1 =' t_2$.

Here we should remember that the terms are called *patterns* by Angluin [3].

DEFINITION 3. We define *well-formed-formulas* as follows:

(1) If P is a predicate and t is a term, then $P(t)$ is an (atomic) formula. If t is a constant string, then $P(t)$ is called a *ground atom*.

(2) If $P, P_1, P_2, \cdots, P_n$ are predicates, and $t, t_1, t_2, \cdots, t_n$ are terms, then $P(t) \leftarrow P_1(t_1) \& P_2(t_2) \& \cdots \& P_n(t_n)$ is a formula. (We regard a formula whose right part is empty, i.e., $n=0$ as an atomic formula $P(t)$.) Formula $P(t)$ is called a *conclusion* and each formula $P_i(t_i)$ on the right side of $\leftarrow$ is called a *premise*.

DEFINITION 4. An *elementary formal system* is a finite set of formulas.

DEFINITION 5. Let θ be any homomorphism from terms to terms. We denote the image of a term t by $t\theta$. If θ maps any constant symbol a to itself, then θ is called a *substitution*. For a formula $F=P(t) \leftarrow P_1(t_1) \& P_2(t_2) \& \cdots \& P_n(t_n)$, we define $F\theta = P(t\theta) \leftarrow P_1(t_1\theta) \& P_2(t_2\theta) \& \cdots \& P_n(t_n\theta)$.

DEFINITION 6. Let $E=\{F_1, F_2, \cdots, F_m\}$ be an elementary formal system. We say that a formula $F=P(t) \leftarrow P_1(t_1) \& P_2(t_2) \& \cdots \& P_{n-1}(t_{n-1})$ is *provable from* E if F satisfies one of the following conditions:

(1) F is in E,

(2) $F=F'\theta$ for some formula F' provable from E and some substitution θ, and

(3) two formulas $P(t) \leftarrow P_1(t_1) \& P_2(t_2) \& \cdots \& P_{n-1}(t_{n-1}) \& P_n(t_n)$ and $P_n(t_n)$ are provable from E.

DEFINITION 7. A *language* $L(E, P)$ defined by an elementary formal system E is the set $\{t \mid t \in \Sigma^*$ and $P(t)$ is provable from $E\}$. When E contains just one predicate, we simply write $L(E)$ instead of $L(E, P)$.

DEFINITION 8. If an elementary formal system E is identical to E' except renam-

ing of variables and rearranging of premises, then we say that E is equivalent to E' and denote it by $E \simeq E'$.

The following proposition is obvious from definitions.

PROPOSITION 1. *If two elementary formal systems E and E' are equivalent, then $L(E, P)=L(E', P)$ for any predicate P.*

EXAMPLE 1.

(1) Let $E=\{P(axbx)\}$. Then $L(E)=\{w \mid w=axbx, x \in \Sigma^*\}$ is the extended language of a pattern $axbx$ [6]. Clearly from this example, the class of languages defined by elementary formal systems is a natural extension of pattern languages.

(2) Let $E=\{P(\varepsilon), P(ax) \leftarrow P(x), Q(bxbxbx) \leftarrow P(x)\}$. Then $L(E, P)=\{a^n \mid n \geqq 0\}$ and $L(E, Q)=\{ba^nba^nba^n \mid n \geqq 0\}$.

3. Inductive Inference from Positive Data

In this section, we make a brief review of inductive inference from positive data, according to Angluin [4].

A class of languages $\boldsymbol{L}=L_1, L_2, \cdots$ is said to be an *indexed family of recursive languages* if there exists a computable function f such that

$$f(i, x)=\begin{cases}1, & \text{if } x \in L_i; \\ 0, & \text{otherwise.}\end{cases}$$

The index i of a language L_i can be considered naturally as a grammar or an autmaton which accepts L_i. In our case, formal systems are used as the indexes of languages.

From here on, the classes of languages are assumed to be an indexed family of recursive languages.

DEFINITION 9. A *complete presentation* of a language L is an infinite sequence $(s_1, t_1), (s_2, t_2), \cdots$ such that t_i is 0 or 1, $\{s \mid s=s_i$ and $t_i=1$ for some $i\}=L$, and $\{s \mid s=s_i$ and $t_i=0$ for some $i\}=\Sigma^*-L$.

DEFINITION 10. A *positive presentation* of a nonempty language L is an infinite sequence of strings $s_1, s_2 \cdots$ such that $\{s \mid s=s_i$ for some $i\}=L$.

DEFINITION 11. *Inductive inference machine* M is an effective procedure that produces an output from an input of finite sequence. $M[\Phi]$ denotes an output produced by an inference machine M from a finite sequence Φ. The output $M[\Phi]$ is called a *guess*. Let $\sigma=s_1, s_2, \cdots$ be an infinite sequence, and denote by $\sigma\langle n\rangle$ a finite subsequence $s_1, \cdots, s_n$ of σ. We say that M *on input* σ *converges to* τ if $M[\sigma\langle n\rangle]$ is defined for all $n \geqq 1$ and there exists an integer N such that $M[\sigma\langle n\rangle]=\tau$ for all $n \geqq N$.

DEFINITION 12. A class of languages $\boldsymbol{L}=L_1, L_2, \cdots$ *is* said to be *inferable from complete (positive) data* if there exists an inference machine M such that M on input σ converges to j with $L_j=L_i$ for any index i and any complete (positive) presentation σ of L_i.

Gold [9] showed that any indexed family of recursive languages is inferable from complete data but is not always inferable from positive data. He also showed that any superfinite class is not inferable from positive data. From his result, we can easily

show that even the class of regular languages is not inferable from positive data. This facts sounds negative for approaches to find natural and interesting applications of inductive inference. Angluin [4], however, showed a theorem characterizing classes inferable from positive data and presented nontrivial and interesting classes including the pattern languages [3]. After her publications, several studies on applications of inductive inference have been made [5, 6, 7, 8, 10, 11].

DEFINITION 13. Let $\boldsymbol{L}=L_1, L_2, \cdots$ be a class of languages. The set T_i is called a *finite tell-tale* of L_i if

(1) T_i is finite,

(2) $T_i \subseteq L_i$, and

(3) $T_i \subseteq L_j \Rightarrow \neg(L_j \subsetneqq L_i)$ for all j.

DEFINITION 14. We say that a class of languages $\boldsymbol{L}=L_1, L_2, \cdots$ satisfies *Condition* 1 if there exists an effective procedure that enumerates all elements in a finite tell-tale of L_i for any i.

THEOREM 2. (Angluin) *A class $L=L_1, L_2, \cdots$ of nonempty languages is inferable from positive data if and only if $\boldsymbol{L}$ satisfies Condition 1.*

DEFINITION 15. We say that a class $\boldsymbol{L}=L_1, L_2 \cdots$ satisfies *Condition 2* if

$$C(S)=\{L \mid S \subseteq L \quad \text{and} \quad L=L_i \text{ for some } i\}$$

is of finite cardinality, for any nonempty finite language S.

COROLLARY 3. (Angluin) *If a class $\boldsymbol{L}=L_1, L_2, \cdots$ satisfies Condition 2, then $\boldsymbol{L}$ is inferable from positive data.*

Although Condition 2 is not necessary for inferability from positive data, it is convenient because it is independent of the indexing. In fact, many classes, including the class of pattern languages of Angluin [3], are shown to be inferable from positive data by using Corollary 3.

4. Restriction of Formal Systems

It is well-known that any recursively enumerable language is representable in an elementary formal system. Therefore we should put some restrictions on elementary formal systems to discuss inferability of languages from positive data. Arikawa discussed restricted formal systems, called simple formal systems [2].

DEFINITION 16. An elementary formal system E is called to be *simple* if each formula in E is of the form

$$P(t(x_1, x_2, \cdots, x_n)) \longleftarrow P_1(x_1) \,\&\, P_2(x_2) \,\&\, \cdots \,\&\, P_n(x_n);$$

where $P, P_1, P_2, \cdots, P_n$ are predicates, $x_1, x_2, \cdots, x_n$ are distinct variables, and $t(x_1, x_2, \cdots, x_n)$ is a term containing at least n variables $x_1, x_2, \cdots, x_n$.

In a simple formal system, the arguments of the premises are distinct variables occurring in the conclusion. Arikawa showed that any context-free language is represented by a simple formal system [2]. Since the class of context-free languages is superfinite and it is not inferable from positive data, we should pay attention to more restricted systems.

DEFINITION 17. A simple formal system E is called to be *primitive* if it contains exactly two formulas of the forms

$$P(t_1) \quad \text{and} \quad P(t_2(x_1, x_2, \cdots, x_n)) \leftarrow P(x_1) \& P(x_2) \& \cdots \& P(x_n).$$

$P(t_1)$ is called a *base step* of E and $P(t_2(x_1, x_2, \cdots, x_n)) \leftarrow P(x_1) \& P(x_2) \& \cdots \& P(x_n)$ is called a *induction step* of E.

Primitive formal systems contain only one predicate. From here on, we always use P to denote the predicate in our primitive formal systems.

DEFINITION 18. Let $X=\{x_1, x_2, \cdots\}$. We define *canonical terms and formulas* as follows:

(1) A term t is canonical if t contains exactly n variables $x_1, \cdots, x_n$ and the leftmost occurrence of x_i is to the left of the leftmost occurrence of x_{i+1} for all $i=1, \cdots, n$.

(2) If t is a canonical term, then an atomic formula $P(t)$ is canonical.

(3) If $t(x_{i_1}, x_{i_2}, \cdots, x_{i_n})$ is a canonical term and $1 \leqq i_1 < i_2 < \cdots < i_n$, then a formula $P(t(x_{i_1}, x_{i_2}, \cdots, x_{i_n})) \leftarrow P(x_{i_1}) \& P(x_{i_2}) \& \cdots \& P(x_{i_n})$ is canonical

We say that a primitive formal system E is *canonical* if each formula in E is canonical. The following proposition is obvious.

PROPOSITION 4. *For any primitive formal system E, there exists a unique canonical primitive formal system E' such that $E' \simeq E$.*

We also restrict substitutions *not to erase variables*. Hereafter we assume that substitutions map each variable to a nonempty term.

EXAMPLE 2. Let the constant alphabet be $\Sigma=\{a, b\}$ and let the variable alphabet be $X=\{x, y, \cdots\}$. The following formal systems are primitive.

(1) $E_1=\{P(axbx),\ P(yx) \leftarrow P(x) \& P(y)\}$.

(2) $E_2=\{P(a),\ P(xx) \leftarrow P(x)\}$.

(3) $E_3=\{P(axby),\ P(xx) \leftarrow P(x)\}$.

E_1 is not canonical. $E_1'=\{P(axbx),\ P(xy) \leftarrow P(x) \& P(y)\} \simeq E_1$ is canonical. $L(E_1)$ contains strings *aaba, abbb, aabaaaba, aabaabbb, abbbaaba, abbbabbb*, $\cdots$. The string *ab* is not in $L(E_1)$ because erasing substitutions are prohibited. E_2 and E_3 are canonical. $L(E_2)=\{a^{2^n} \mid n \geqq 0\}$. $L(E_2)$ is not context-free and not a pattern languages. $L(E_3)=L(\{P(axby)\})$ is the language of a pattern *axby* [3]. In E_3, the induction step $P(xx) \leftarrow P(x)$ is easily shown to be redundant.

If a formula $P(x) \leftarrow P(x)$ is contained in a primitive formal system E then it is always redundant in the sense that the language $L(E)$ is equal to that represented by only the base step of E. If the induction step of E is redundant, then $L(E)$ is a pattern language. Thus the class of languages defined by primitive formal systems is a proper superclass of the class of pattern languages.

5. Inductive Inference of Formal Systems from Positive Data

In this section we show that the class of languages representable in the primitive formal systems is inferable from positive data.

First, we show, as an exercise, the inferability from positive data in case the base cases are ground atoms. We slightly modify Condition 2 in Definition 15 and

show that modified condition is still sufficient for inferability from positive data.

DEFINITION 19. We say that a class $\boldsymbol{L}=L_1, L_2, \cdots$ satisfies *Condition 2'* if

$$C(w_1, w_2)=\{L \mid w_1 \in L, w_2 \in L, L=L_i \text{ for some } i\}$$

is of finite cardinality, for any distinct strings w_1 and w_2.

LEMMA 5. *If a class $\boldsymbol{L}=L_1, L_2, \cdots$ of nonempty languages satisfies Condition 2', then $\boldsymbol{L}$ is inferable from positive data*

PROOF: We show that Condition 1 follows from Condition 2'. Let L be any class of nonempty languages satisfying Condition 2'. For simplicity, we first assume that $L_j \subsetneqq L_i$ is computable from j and i. We define procedure Q as follows. Let i be the input to Q. Start at Stage 0.

Stage 0: If L_i contains only one element s_1, then output s_1. Otherwise, output s_1 and another element s_1' in L_i, set $T_1=\{s_1, s_1'\}$, and go to Stage 1.

Stage n: ($n \geqq 1$) Find an integer j such that $T_n \subseteq L_j \subsetneqq L_i$. If such an integer j is found, then output any element s_n in $L_i - L_j$, set $T_{n+1}=T_n \cup \{s_n\}$, and go to Stage $n+1$.

We show Q enumerates all elements in a finite tell-tale of L_i. Assume that L_i contains only one element. Then L_i itself is a finite tell-tale and Q outputs the element of L_i. Assume that L_i contains two or more elements. In this case, Q outputs two elements s_1 and s_2 in L_i and go to Stage N, Q fails to find j, and never go to Stage $N+1$, because the class $\boldsymbol{L}$ satisfies Condition 2', that is, the number of languages s_1 containing s_1 and s_2 is finite. Therefore, T_N is a finite tell-tale of L_i.

We can remove our assumption that $L_j \subsetneqq L_i$ is computable from j and i, by modifying Stage n($n \geqq 1$) as follows. Let $w_1, w_2, \cdots$ be an effective enumeration of Σ^* and $L_i^{(m)}=L_i \cap \{w_1, \cdots, w_m\}$.

Stage n: ($n \geqq 1$) Find a pair (j, m) of integers such that $T_n \subseteq L_j$ and $L_j^{(m)} \subsetneqq L_i^{(m)}$. If such a pair is found, then output any element s_n in $L_i^{(m)} - L_j^{(m)}$, set $T_{n+1}=T_n \cup \{s_n\}$, and go to Stage $n+1$.

$L_i^{(m)}$ is a computable finite set, and hence decision $L_j^{(m)} \subsetneqq L_i^{(m)}$ is computable. We can effectively enumerate all pairs of integers. Therefore Q effectively enumerates elements in a finite tell-tale of L_i. □

THEOREM 6. *The class of languages defined by primitive formal systems base steps of which are ground atoms is inferable from positive data.*

PROOF: Let $E=\{P(w), P(t(x_1, x_2, \cdots, x_n)) \leftarrow P(x_1) \& P(x_2) \& \cdots \& P(x_n)\}$. be a primitive formal system and w be a nonempty constant string. Assume that distinct strings s_1 and s_2 are contained in $L(E)$. Clearly,

$$|w| \leqq |s_1| \quad \text{and} \quad |w| \leqq |s_2|, \tag{1}$$

Since the base step $P(w)$ is a ground atom and only one formula $P(w)$ is provable from the base step, the induction step of E is used by at least one of the proofs of $P(s_1)$ and $P(s_2)$. We may assume, without loss of generality, that $P(s_1)$ is proved by using the induction step $P(t(x_1, x_2, \cdots, x_n)) \leftarrow P(x_1) \& P(x_2) \& \cdots \& P(x_n)$. Then, there exists a non-erasing substitution that maps $t(x_1, x_2, \cdots, x_n)$ to s_1. Therefore,

$$|t(x_1, x_2, \cdots, x_n)| \leqq |s_1|, \tag{2}$$

From Proposition 1 and Proposition 4, we have

$$C(s_1, s_2) = \{L(E) \mid s_1 \in L(E),\ s_2 \in L(E),$$
$$E \text{ is a primitive formal system and}$$
$$\text{the base step of } E \text{ is a ground atom}\}$$

$$= \{L(E) \mid s_1 \in L(E),\ s_2 \in L(E),$$
$$E \text{ is a canonical primitive formal system, and}$$
$$\text{the base step of } E \text{ is a ground atom}\}.$$

From (1) and (2), the number of canonical primitive formal systems E such that $L(E)$ contains both s_1 and s_2 is finite. Therefore, the class of languages defined by primitive formal systems, base steps of which are ground atoms, satisfies Condition 2', and hence by Lemma 5, it is inferable from positive data. □

LEMMA 7. *Let E be a primitive formal systems and $S_0(E)$ be the set of the shortest strings in $L(E)$. Then*

$$S_0(E) = L(\{P(t)\}) \cap \Sigma^{|t|},$$

where $P(t)$ is the base step of E.

PROOF: Let s be one of the shortest strings in $L(E)$. Since $P(s)$ can be proved by the base step $P(t)$, we have $S_0(E) \subseteq L(\{P(t)\})$. The length of the shortest strings in $L(\{P(t)\})$ is equal to $|t|$, because substitution for variables are restricted to nonempty terms. Thus we have $S_0(E) = L(\{P(t)\}) \cap \Sigma^{|t|}$. □

Here we should remember that the set of the shortest strings in a pattern language is a finite tell-tale [3].

LEMMA 8. *Let E_1 and E_2 be any primitive formal systems, and let $P(t_1)$ and $P(t_2)$ be the base steps of E_1 and E_2, respectively. Then*

$$S_0(E_1) \subseteq L(E_2) \subseteq L(E_1) \Rightarrow t_1 = {}' t_2,$$

PROOF: $S_0(E_1) \subseteq L(E_2)$ implies $|t_1| \geqq |t_2|$. $L(E_2) \subseteq L(E_1)$ implies $|t_2| \geqq |t_1|$. Therefore $|t_1| = |t_2|$ and $S_0(E_1) = S_0(E_2)$. Thus, t_1 and t_2 are equivalent. □

Now we state and prove our main theorem. The proof closely resembles that of the main theorem on unions of two pattern languages [7].

THEOREM 9. *The class of languages defined by primitive formal systems is inferable from positive data.*

PROOF: We give a proof by showing that the class of languages defined by primitive formal systems satisfies Condition 1. Here it should be noticed that both Condition 2 and Condition 2' are violated.

Consider the following procedure Q. The input to Q is a primitive formal system E, instead of an index of a language. Let $P(t)$ be the base step of E and let $w_1, w_2 \cdots$ be an effective enumeration of all nonempty constant strings.

```
procedure Q(E)
    begin
        output all elements in S0(E)
        T:=S0(E)
        i:=1
        while wi∉L(E)−L({P(t)}) do i:=i+1
        output wi
        T:=T∪{wi}
        B:={F | F=P(t'(x1, x2, ⋯, xn))←P(x1) & P(x2) & ⋯ & P(xn),
              F is a canonical formula,
              |t'(x1, x2, ⋯, xn)| ≦ |wi|, and
              T⊆L({P(t), F})}
        for j:=1 to ∞ do
            if wj∈L(E) then
                begin
                    B':={F∈B | wj∉L({P(t), F})}
                    if B'≠∅ then
                        begin
                            output wj
                            T:=T∪{wj}
                            B:=B−B'
                        end
                end
    end
```

Since $S_0(E)$ is computable from E by Lemma 7, the effectiveness of Q is obvious. We show that Q enumerates a finite tell-tale of $L(E)$. Note that the **while** loop checks the redundancy of the induction step. We consider two cases depending on whether the induction step of E is redundant or not.

Case 1. If the induction step is redundant, then the **while** loop never terminates. The set enumerated by Q is $S_0(E)$. Since the induction step of E is redundant

$$L(E)=L(\{P(t)\}).$$

Assume that there exists another primitive formal system E' such that $S_0(E)\subseteq L(E')$. By Lemma 8, the base step of E' is equivalent to $P(t)$ and clearly

$$L(\{P(t)\})\subseteq L(E').$$

Therefore $L(E')$ is not properly contained in $L(E)$, that is, $S_0(E)$ is a finite tell-tale of $L(E)$.

Case 2. If the induction step is not redundant, then the **while** loop eventually terminates and a string w_i in $L(E)-L(\{P(t)\})$ is found. Assume that there exists another canonical primitive formal system E' such that $S_0(E)\cup\{w_i\}\subseteq L(E')\subsetneq L(E)$. By Lemma 8, the base step of E' is equivalent to $P(t)$. Let $F=P(t'(x_1, x_2, \cdots, x_n))\leftarrow P(x_1)\,\&\,P(x_2)\,\&\cdots\&\,P(x_n)$ be the induction step of E'. Since w_i is not in $L(\{P(t)\})$, F is not redundant, that is, $P(w_i)$ is proved by using F from E', and hence there exists a non-erasing substitution that maps $t'(x_1, x_2, \cdots, x_n)$ to w_i. Therefore

$$|t'(x_1, x_2, \cdots, x_n)| \leqq |w_i|,$$

and any canonical formula F' such that

$$S_0(E) \cup \{w_i\} \subseteq L(\{P(t), F'\}) \subsetneqq L(E)$$

is always contained in B calculated before the **for** loop. The number of formulas in B is finite because the number of premises of any formula in B is bounded by $|w_i|$. For each formula in B such that $L(\{P(t), F\}) \subsetneqq L(E)$, the **for** loop will find a string w_j in $L(E) - L(\{P(t), F\})$, output w_j, and remove F from B. Since B is finite, Q outputs finitely many strings. From our observation, it is clear that the set enumerated by Q is a finite tell-tale of $L(E)$. □

6. Concluding Remarks

We have discussed inductive inference of formal systems from positive data. Since recursively enumerable languages are definable by Smullyan's elementary formal systems [1], we have paid attention to restricted formal systems. The class of languages represented by Arikawa's simple formal systems [2] is a indexed family of recursive languages, and is located between the classes of context-free languages and context-sensitive languages. Clearly from Gold's theorem [9], his class is not inferable from positive data.

The most important class inferable from positive data should be the class of pattern languages introduced by Angluin [3]. The main purpose of the present work is to find a class of languages inferable from positive data that is a natural extension of Angluin's class. As we have seen, we have succeeded to define a superclass of Angluin's pattern languages by using restricted formal systems, named primitive formal systems, and to show their inferability from positive data. It should be an important and interesting problem to find more general formal systems that are still inferable from positive data.

The author studied Angluin's pattern languages from the viewpoint of practical applications of inductive inference [5, 6, 7, 8]. The main subject was computational complexity of inductive inference rather than inferability from positive data. It should be another important problem to consider practical application of inductive inference of formal systems.

We can regard formulas in elementary formal systems as Horn formulas in logic programming [12] over strings in which only the concatenation is used as a function. Shapiro [13] studied inductive inference of theories and showed that his result is applicable to Prolog program synthesis and debugging. Another discussion will be needed to apply Shapiro's result to inductive inference of formal systems, because the unifications in formal systems are quite different from those in the usual logic programming. For example, two terms $f(a, f(X, b))$ and $f(f(a, X), b)$ are not unifiable in the usual sense, but they are equivalent to each other when f is a concatenation, and therefore they must be unified in our formal systems.

Acknowledgement

The motivation of the present work is due to Prof. Setsuo Arikawa. The author is grateful to him for his constant support and encouragement.

References

[1] SMULLYAN, R. M.: *Theory of Formal Systems*, Princeton University Press, Princeton, New Jersey, (1961).

[2] ARIKAWA, S.: *Elementary Formal Systems and Formal Languages-Simple Formal Systems*, Memoirs of the Faculty of Science, Kyushu University, Series A, Mathematics, **24**, (1970), 47-75.

[3] ANGLUIN, D.: *Finding Common Patterns to a Set of Strings*, In "Proceedings of the 11 th Annual Symposium on Theory of Computing," (1979), 130-141.

[4] ANGLUIN. D.: *Inductive Inference of Formal Languages from Positive Data*, Information and Control **45**, (1980), 117-135.

[5] SHINOHARA, T.: *Polynomial Time Inference of Pattern Languages and Its Application*, in "Proceedings of the 7 th IBM Symposium on Mathematical Foundations of Computer Science," (1982), 191-209.

[6] SHINOHARA, T.: *Polynomial Time Inference of Extended Regular Pattern Languages*, in "Proceedings of RIMS Symposia on Software Science and Engineering, Kyoto, 1982," Lecture Notes in Computer Science **147**, Spinger-Verlag, (1983), 115-127.

[7] SHINOHARA, T.: *Inferring Unions of Two Pattern Languages*, Bulletin of Informatics and Cybernetics, **20**, (1983), 83-88.

[8] SHINOHARA, T. and ARIKAWA, S.: *Learning Data Entry System: An Application of Inductive Inference of Pattern Languages*, Research Report 102, Research Institute of Fundamental Information Science, Kyushu University, (1983).

[9] GOLD, E. M.: *Language Identification in the Limt*, Information and Control **10**, (1967), 447-474.

[10] NIX, R. P.: *Editing By Example*, Research Report 280, Department of Computer Science, Yale University, (1983).

[11] JANTKE, K. P.: *Polynomial Time Inference of General Pattean Languages*, in "Proceedings of STACS '84, Paris," Lecture Notes in Computer Science **166**, Springer-Verlag, (1984), 314-325.

[12] LLOYD, J. W.: *Foundations of Logic Programming*, Springer-Verlag, (1984).

[13] SHAPIRO, E.: *Inductive Inference of Theories from Facts*, Technical Report 194, Department of Computer Science, Yale University, (1981).

Communicated by S. Arikawa
Received August 27, 1985