社団法人　情報処理学会　研究報告
IPSJ SIG Technical Report

2005－AL－102（1）
2005／9／16

正の例から極限同定可能な言語クラスを拡張する統一的アルゴリズム

若月　光夫 [†]　　富田　悦次 [†]　　山田　　剛 [†]

[†] 電気通信大学 電気通信学部 情報通信工学科　〒 182–8585 東京都調布市調布ケ丘 1–5–1
E-mail: [†]{wakatuki,tomita}@ice.uec.ac.jp

あらまし　ある種の言語クラスを準同型写像で変換して得られるような，比較的単純な言語クラスの拡張にあたる幾つかのクラスが正の例から極限同定可能であることが近年報告されている．本稿では，これらの言語クラスを統一的に扱えるような枠組みを提案する．ここで，言語クラスの拡張は基となる言語クラス $\mathcal{L}$ と有限記号列集合のクラス $\mathcal{X}$ に対して定義され，拡張された言語クラスは $\mathcal{C}(\mathcal{L},\mathcal{X})$ で表される．この言語クラス $\mathcal{C}(\mathcal{L},\mathcal{X})$ が正の例から極限同定可能であるための十分条件を前提として与え，$\mathcal{C}(\mathcal{L},\mathcal{X})$ に対する正の例からの極限同定アルゴリズム $\mathcal{A}_{\mathcal{C}(\mathcal{L},\mathcal{X})}$ を提案する．また，これを適用した具体例として，$\mathcal{C}(\mathcal{L},\mathcal{X})$ の幾つかの真部分クラスが正の例から多項式時間極限同定可能であることを示す．

A Unified Algorithm for Extending Classes of Languages Identifiable in the Limit from Positive Data

Mitsuo WAKATSUKI[†], Etsuji TOMITA[†], and Go YAMADA[†]

[†]Department of Information and Communication Engineering, Faculty of Electro-Communications,
The University of Electro-Communications　Chofugaoka 1–5–1, Chofu, Tokyo 182-8585, Japan
E-mail: [†]{wakatuki,tomita}@ice.uec.ac.jp

Abstract.　We are concerned with a *unified* algorithm for extending classes of languages identifiable in the limit from positive data. Let $\mathcal{L}$ be a class of languages to be based on and $\mathcal{X}$ a class of finite subsets of strings. The extended class of $\mathcal{L}$, denoted by $\mathcal{C}(\mathcal{L},\mathcal{X})$, is defined by these $\mathcal{L}$ and $\mathcal{X}$. Then we give a sufficient condition for $\mathcal{C}(\mathcal{L},\mathcal{X})$ to be identifiable in the limit from positive data and present an identification algorithm for it. Furthermore, we show that some proper subclasses of $\mathcal{C}(\mathcal{L},\mathcal{X})$ are polynomial time identifiable in the limit from positive data in the sense of Yokomori.

1　Introduction

In the study of inductive inference of formal languages, Gold [4] defined the notion of *identification in the limit* and showed that the class of languages containing all finite sets and one infinite set, which is called a *superfinite* class, is not identifiable in the limit from *positive data*. This means that even the class of regular languages is not identifiable in the limit from positive data. Angluin [1] has given several conditions for a class of languages to be identifiable in the limit from positive data, and has presented some examples of identifiable classes. She has also proposed subclasses of regular languages called k-reversible languages for each $k \geq 0$, and has shown that these classes are identifiable in the limit from positive data, requiring a polynomial time for updating conjectures [2].

From the practical point of view, the inductive inference algorithm must have a good *time efficiency* in addition to running with only positive data. One may define the notion of *polynomial time identification in the limit* in various ways. Pitt [8] has proposed a reasonable definition for polynomial time identifiability in the limit. By making a slight modification of his definition, Yokomori [9] has proposed another definition for polynomial time identifiability in the limit from positive data, and has proved that a class of languages accepted by strictly deterministic automata (SDAs) [9], which is a proper subclass of regular languages, is polynomial time identifiable in the limit from positive data.

An SDA is an extended deterministic finite automaton, which is intuitively a state transition graph in which the set X of labels for edges is a finite subset of strings over an alphabet Σ, that satisfies the following conditions: for any

x in X, there uniquely exists an edge (a pair of states) whose label is x, and for any distinct labels x_1, x_2 in X, the first symbol of x_1 differs from that of x_2. This SDA can be also represented by a pair (M, φ) of a corresponding deterministic finite automaton M and a homomorphism $\varphi : \Sigma'^* \to \Sigma^*$ such that $X = \varphi(\Sigma')$ for some alphabet Σ', where the language accepted by M is in the class of Szilard languages of linear grammars [7]. That is, the class of languages accepted by SDAs is the extended class of Szilard languages of linear grammars. In a similar way to this, some kind of language classes can be extended.

In this paper, we are concerned with a unified algorithm for extending classes of languages identifiable in the limit from positive data. Let $\mathcal{L}$ be a class of languages over Σ' to be based on and $\mathcal{X}$ a class of finite subsets of strings over Σ, where there exists a morphism $\varphi : \Sigma'^* \to X^*$ for some $X \in \mathcal{X}$. The extended class of $\mathcal{L}$, denoted by $\mathcal{C}(\mathcal{L}, \mathcal{X})$, is defined by these $\mathcal{L}$ and $\mathcal{X}$. Kobayashi and Yokomori [6] proved that for each $k \geq 0$, a class $\mathcal{C}(\mathcal{R}ev_k, \mathcal{X}_0)$ of languages, where $\mathcal{R}ev_k$ is a class of k-reversible languages and $\mathcal{X}_0$ is a class of codes [3], is identifiable in the limit from positive data. However, they have not shown the identification algorithm for $\mathcal{C}(\mathcal{R}ev_k, \mathcal{X}_0)$, and it is still unknown whether the time complexity of the algorithm is polynomial in the sense of Yokomori [9]. Here we give a sufficient condition for $\mathcal{C}(\mathcal{L}, \mathcal{X})$ to be identifiable in the limit from positive data and present an identification algorithm for it. Furthermore, we show that some proper subclasses of $\mathcal{C}(\mathcal{R}ev_k, \mathcal{X}_0)$ are polynomial time identifiable in the limit from positive data.

2 Definitions

2.1 Basic Definitions and Notation

We assume that the reader is familiar with the basics of automata and formal language theory. For the definitions and notation not stated here, see, e.g., [5].

A *semigroup* consists of a set S with a binary associative operation defined on S. A *monoid* M is a semigroup which possesses a two-sided identity, where the identity element is denoted by ε_M or simply by ε. A *morphism* from a monoid M into a monoid N is a function $\varphi : M \to N$ which satisfies, for all $m_1, m_2 \in M$, $\varphi(m_1 m_2) = \varphi(m_1)\varphi(m_2)$, and furthermore $\varphi(\varepsilon_M) = \varepsilon_N$.

An *alphabet* Σ is a finite set of symbols. For any finite set S of finite-length strings over Σ, we denote by S^* (respectively, S^+) the set of all finite-length strings obtained by concatenating zero (one, resp.) or more elements of S, where the concatenation of strings u and v is simply denoted by uv. Note that the set S^* (respectively, S^+) is the submonoid (subsemigroup, resp.) generated by S. In particular, Σ^* denotes the set of all finite-length strings over Σ. The string of length 0 (the empty string) is denoted by ε. We denote by $|w|$ the length of a string w and by $|S|$ the cardinality of a set S. A *language* over Σ is any subset L of Σ^*. For a string $w \in \Sigma^*$, $\text{alph}(w)$ denotes the set of symbols appearing in w. For a language $L \subseteq \Sigma^*$, let $\text{alph}(L) = \cup_{w \in L} \text{alph}(w)$.

2.2 Polynomial Time Identification in the Limit from Positive Data

In this paper, we adopt Yokomori's definition in [9] for the notion of polynomial time identification in the limit from positive data.

For any class of languages to be identified, let $\mathcal{R}$ be a *class of representations* for a class of languages. Instances of such representations are automata, grammars, and so on. Given an r in $\mathcal{R}$, $L(r)$ denotes the language represented by r. A *positive presentation* of $L(r)$ is any infinite sequence of data such that every $w \in L(r)$ occurs at least once in the sequence and no other string not in $L(r)$ appears in the sequence. Each element of $L(r)$ is called a *positive example* (or simply, *example*) of $L(r)$.

Let r be a representation in $\mathcal{R}$. An algorithm $\mathcal{A}$ is said to *identify r in the limit from positive data* iff $\mathcal{A}$ takes any positive presentation of $L(r)$ as an input, and outputs an infinite sequence of representations in $\mathcal{R}$ such that there exist r' in $\mathcal{R}$ and $j > 0$ so that for all $i \geq j$, the i-th conjecture (representation) r_i is identical to r' and $L(r') = L(r)$. A class $\mathcal{R}$ is *identifiable in the limit from positive data* iff there exists an algorithm $\mathcal{A}$ that, for any r in $\mathcal{R}$, identifies r in the limit from positive data.

Let $\mathcal{A}$ be an algorithm for identifying $\mathcal{R}$ in the limit from positive data. Suppose that after examining i examples, the algorithm $\mathcal{A}$ conjectures some r_i. We say that $\mathcal{A}$ makes an *implicit error of prediction* at step i if r_i is not consistent with the $(i+1)$-st example w_{i+1}, i.e., if $w_{i+1} \notin L(r_i)$.

Definition 1.(Yokomori [9], pp.157-158, Definition 2) A class $\mathcal{R}$ is *polynomial time identifiable* *in the limit from positive data* iff there exists an algorithm $\mathcal{A}$ for identifying $\mathcal{R}$ in the limit from positive data with the property that there exist polynomials p and q such that for any n, for any r of size n, and for any positive presentation of $L(r)$, the time used by $\mathcal{A}$ between receiving the i-th example w_i and outputting the i-th conjecture r_i is at most $p(n, \sum_{j=1}^{i} |w_j|)$, and the number of implicit errors of prediction made by $\mathcal{A}$ is at most $q(n, l)$, where the size of r is the length of a description for r and $l = \mathrm{Max}\{|w_j| \mid 1 \leq j \leq i\}$. □

3 Language Classes Extended by Using Codes

In this section, we introduce language classes extended by using codes.

Definition 2. Let $w \in \Sigma^*$ and $X, Y \subseteq \Sigma^*$. Any sequence $(v_1, v_2, \ldots, v_n) \in (\Sigma^*)^n$ $(n \geq 0)$ such that $w = v_1 v_2 \cdots v_n$ is called a *factorization* of w. Moreover, any sequence $(x_1, x_2, \ldots, x_n) \in X^n$ such that $w = x_1 x_2 \cdots x_n$ is called an X-*factorization* of w. If w has an X-factorization, i.e., $w \in X^*$, we say that X *can factorize* w. Moreover, if any string in Y has an X-factorization, i.e., $Y \subseteq X^*$, we say that X *can factorize* Y. □

Definition 3. A finite set X over Σ is said to be a *finite factorizing set* iff, for any x in X, $|x|$ is finite and $|x| \geq 1$. □

The next lemma follows from Definitions 2 and 3.

Lemma 1. Let $\mathcal{X}$ be a class of finite factorizing sets over Σ and $S \subseteq \Sigma^*$. For any $X \in \mathcal{X}$, it holds that X can factorize $S \cup \{\varepsilon\}$ iff X can factorize $S - \{\varepsilon\}$.
Proof: It can be proved by using the fact that X can factorize ε for any X. □

Definition 4. A finite factorizing set X over Σ is said to be *ambiguous* iff there exists $w \in X^+$ which has at least two distinct X-factorizations. Otherwise, it is said to be *unambiguous*. Moreover, a class $\mathcal{X}$ of finite factorizing sets over Σ is said to be *ambiguous* iff there exists $X \in \mathcal{X}$ which is ambiguous. Otherwise, it is said to be *unambiguous*. □

If a finite factorizing set X over Σ is unambiguous, X is also called a *code* ([3], p.38). Therefore, a class of unambiguous finite factorizing sets is a class of codes. Note that a code never contains the empty string ε. It is clear that any subset of a code is a code. In particular, the empty set $\emptyset$ is a code.

Definition 5. Let $\mathcal{X}$ be a class of finite factorizing sets over Σ and $X \in \mathcal{X}$ a finite factorizing set which can factorize S for some nonempty subset S of Σ^*. X is said to be the *coarsest* finite factorizing set in $\mathcal{X}$ which can factorize S iff, for any $X' \in \mathcal{X}$ which can factorize S, X' can factorize X. □

From Definitions 4 and 5, we have the following lemma.

Lemma 2. Let $\mathcal{X}$ be a class of codes (i.e., a class of unambiguous finite factorizing sets) over Σ and S a nonempty subset of Σ^*. Then, there exists at most one coarsest code in $\mathcal{X}$ which can factorize S.
Proof: Suppose for the sake of contradiction that there exist two distinct coarsest codes X_1, X_2 in $\mathcal{X}$ which can factorize S. Since X_1 is the coarsest code which can factorize S, it holds that $S \subseteq X_1^*$ and $X_1 \subseteq X_2^*$ from Definition 5. Similarly, it holds that $S \subseteq X_2^*$ and $X_2 \subseteq X_1^*$. By symmetry, we may assume that $X_1 - X_2 \neq \emptyset$ from the assumption that $X_1 \neq X_2$. For any $x \in X_1 - X_2$, it holds that $x \in X_2^+ - X_2$ since $X_1 \subseteq X_2^*$ and $\varepsilon \notin X_1$. Therefore, x has an X_2-factorization $(v_1, v_2, \ldots, v_n)$ such that $n \geq 2$ and $x = v_1 v_2 \cdots v_n$ with $v_i \in X_2$ for $1 \leq i \leq n$. For every $v_i \in X_2$ with $1 \leq i \leq n$, it holds that $v_i \in X_1^+$ since $X_2 \subseteq X_1^*$ and $\varepsilon \notin X_2$. Thus, x has two distinct X_1-factorizations, which is a contradiction since X_1 is unambiguous. □

In Lemma 2, if $S \subseteq X^*$ for some code $X \in \mathcal{X}$, then there exists the coarsest code $X_S \in \mathcal{X}$ which can factorize S (i.e., $S \subseteq X_S^* \subseteq X^*$).

Berstel and Perrin [3] proved the following proposition and corollary from the definition of a code. They are useful for our later discussion.

Proposition 1.(Berstel and Perrin [3], p.38, Proposition 1.1) If a subset X of Σ^* is a code, then any morphism $\varphi : \Sigma'^* \rightarrow \Sigma^*$ which induces a bijection of some alphabet Σ' onto X is injective. Conversely, if there exists an injective morphism $\varphi : \Sigma'^* \rightarrow \Sigma^*$ such that $X = \varphi(\Sigma')$, then X is a code. □

Corollary 1.(Berstel and Perrin [3], pp.39-40, Corollary 1.2) Let $\varphi : \Sigma_1^* \rightarrow \Sigma_2^*$ be an injective morphism. If X is a code over Σ_1, then $\varphi(X)$ is a code over Σ_2. If Y is a code over Σ_2, then $\varphi^{-1}(Y)$ is a code over Σ_1. □

Such an injective morphism φ in Proposition 1 is called a *coding morphism* for X. In Corollary 1, if $Y = \varphi(X)$, then it holds that $\varphi^{-1}(\varphi(X)) = X$.

Let $\mathcal{L}$ be a class of some languages over Σ' and $\mathcal{X}$ a class of some finite factorizing sets over Σ. Now we define a new class of languages over Σ, denoted by $\mathcal{C}(\mathcal{L}, \mathcal{X})$, using these $\mathcal{L}$ and $\mathcal{X}$ as follows.

Definition 6. A class of languages denoted by $\mathcal{C}(\mathcal{L}, \mathcal{X})$ over Σ is defined as the class obtained by the following procedure: (1) For every language $L \in \mathcal{L}$ over Σ', let $\Sigma'_L = \mathrm{alph}(L)$ $(\subseteq \Sigma')$. Note that Σ'_L is a code over Σ'. (2) For every finite factorizing set $X \in \mathcal{X}$ over Σ such that $|X| = |\Sigma'_L|$, define a bijection φ of Σ'_L onto X. (3) For each $L \in \mathcal{L}$ and $\varphi : \Sigma'^* \to \Sigma^*$ such that $X = \varphi(\Sigma'_L)$, a language $\varphi(L) \in \mathcal{C}(\mathcal{L}, \mathcal{X})$ over Σ is defined as $\varphi(L) = \{\varphi(w) \in X^* \mid w \in L\}$. □

Let L_0 be a language over Σ and Y a code over Σ which can factorize L_0. That is, it holds that $L_0 \subseteq Y^* \subseteq \Sigma^*$. Let $\varphi : \Sigma'^* \to \Sigma^*$ be an injective morphism. Since Y is a code over Σ, it holds that $\varphi^{-1}(Y)$ is a code over Σ' from Corollary 1. Since Y can factorize L_0, it holds that $\varphi^{-1}(L_0) \subseteq \varphi^{-1}(Y^*) = [\varphi^{-1}(Y)]^* \subseteq \Sigma'^*$. Therefore, the code $\varphi^{-1}(Y)$ can factorize the language $\varphi^{-1}(L_0)$.

Definition 7. Let $\mathcal{L}$ be a class of some languages over Σ and $\Sigma' \subseteq \Sigma$. We say that the class $\mathcal{L}$ *is closed under inverse coding morphism* iff, for any $L \in \mathcal{L}$ and any code $X \subseteq \Sigma^*$ which can factorize L, it holds that $\varphi^{-1}(L) \in \mathcal{L}$, where $\varphi : \Sigma'^* \to \Sigma^*$ is a coding morphism such that $X = \varphi(\Sigma')$. □

Hereafter, we are only concerned with a class $\mathcal{C}(\mathcal{L}, \mathcal{X})$ of languages that satisfies the following conditions 1 and 2.

Condition 1. The class $\mathcal{L}$ of languages satisfies the following conditions: (1) $\mathcal{L}$ is closed under inverse coding morphism, and (2) $\mathcal{L}$ is identifiable in the limit from positive data. □

Condition 2. The class $\mathcal{X}$ of finite factorizing sets over Σ satisfies the following conditions: (1) $\mathcal{X}$ is unambiguous. (That is, $\mathcal{X}$ is a class of codes.) (2) For any positive presentation of S such that $S \subseteq X^+$ for some $X \in \mathcal{X}$, there exists an algorithm for identifying the coarsest code $X_S \in \mathcal{X}$ which can factorize S in the limit. (This algorithm is called an identification algorithm for the coarsest code in $\mathcal{X}$ in the limit from positive data.) □

4 Identification Algorithm

Let $\mathcal{A}_{\mathcal{L}}$ be an identification algorithm in Condition 1 and $\mathcal{A}_{\mathcal{X}}$ an identification algorithm in Condition 2.

The flow of the algorithm $\mathcal{A}_{\mathcal{L}}$ can be written as follows, where the function CONSTRUCT(r', w') receives a positive example w' of the target language L' and a representation r' for a language in $\mathcal{L}$ as input, and outputs an updated representation for a language in $\mathcal{L}$ obtained by modifying r' from w'.

Identification Algorithm $\mathcal{A}_{\mathcal{L}}$
Input: a positive presentation $w'_1, w'_2, \ldots$ of a target language L' in $\mathcal{L}$
Output: a sequence of representations r'_1, r'_2, …, where r'_i $(i \geq 1)$ is a representation for a language $L(r'_i)$ in $\mathcal{L}$
Procedure

```
begin
  $S'_0 := \emptyset$;   $\Sigma'_0 := \emptyset$;
  initialize $r'_0$ so that $L(r'_0) = \emptyset$;
  $i := 1$;
  repeat (forever)
    read the next positive example $w'_i$;
    $S'_i := S'_{i-1} \cup \{w'_i\}$;   $\Sigma'_i := \Sigma'_{i-1} \cup \mathrm{alph}(w'_i)$;
    if $w'_i \in L(r'_{i-1})$ then $r'_i := r'_{i-1}$
    else $r'_i :=$ CONSTRUCT$(r'_{i-1}, w'_i)$ fi
    output $r'_i$;
    $i := i + 1$
end
```

Moreover, the flow of the algorithm $\mathcal{A}_{\mathcal{X}}$ can be written as follows, where the function UPDATE(X', w') receives a positive example $w' \in X^+$ for the target code $X \in \mathcal{X}$ and a code $X' \in \mathcal{X}$ as input, and outputs the coarsest code in $\mathcal{X}$ which can factorize $X' \cup \{w'\}$.

Identification Algorithm $\mathcal{A}_{\mathcal{X}}$
Input: a positive presentation $w_1, w_2, \ldots$ of X^+, where $X \in \mathcal{X}$ is a target code such that $X = \varphi(\Sigma')$ for some Σ'
Output: a sequence of the coarsest codes X_1, $X_2, \ldots$, where X_i $(i \geq 1)$ can factorize $\{w_1, w_2, \ldots, w_i\}$
Procedure

```
begin
  $S_0 := \emptyset$;   $\Sigma_0 := \emptyset$;   $X_0 := \emptyset$;
  $i := 1$;
  repeat (forever)
    read the next positive example $w_i$;
    $S_i := S_{i-1} \cup \{w_i\}$;   $\Sigma_i := \Sigma_{i-1} \cup \mathrm{alph}(w_i)$;
```

```
        $X_i$ := UPDATE($X_{i-1}, w_i$);
        **output** $X_i$;
        $i := i + 1$
    **end**
```

Now we present an identification algorithm $\mathcal{A}_{\mathcal{C}(\mathcal{L},\mathcal{X})}$ for a class $\mathcal{C}(\mathcal{L},\mathcal{X})$ of languages that satisfies Conditions 1 and 2. This algorithm is an extended version of the above algorithms $\mathcal{A}_{\mathcal{L}}$ and $\mathcal{A}_{\mathcal{X}}$. The algorithm $\mathcal{A}_{\mathcal{C}(\mathcal{L},\mathcal{X})}$ is given in the following.

Identification Algorithm $\mathcal{A}_{\mathcal{C}(\mathcal{L},\mathcal{X})}$
Input: a positive presentation $w_1, w_2, \ldots$ of a target language in $\mathcal{C}(\mathcal{L},\mathcal{X})$
Output: a sequence of pairs $(r_1, X_1), (r_2, X_2), \ldots$ such that $\varphi_i(L(r_i)) \in \mathcal{C}(\mathcal{L},\mathcal{X}) (i \geq 1)$, where r_i is a representation for a language $L(r_i)$ and X_i is a code such that $X_i = \varphi_i(\Sigma'_i)$ for a coding morphism $\varphi_i : \Sigma'^*_i \to \Sigma^*$
Procedure

```
begin
  $S_0 := \emptyset$;  $\Sigma_0 := \emptyset$;  $S'_0 := \emptyset$;  $\Sigma'_0 := \emptyset$;
  $X_0 := \emptyset$;
  initialize $\varphi_0$ so that $\varphi_0(\Sigma'_0) = X_0$;
  initialize $r_0$ so that $L(r_0) = \emptyset$;
  $i := 1$;
  read the next positive example $w_i$;
  while $w_i = \varepsilon$ do
    $S_i := \{\varepsilon\}$;  $\Sigma_i := \emptyset$;
    $S'_i := \{\varepsilon\}$;  $\Sigma'_i := \emptyset$;
    $X_i := \emptyset$;  $\varphi_i := \varphi_{i-1}$;
    if $L(r_{i-1}) = \{\varepsilon\}$ then $r_i := r_{i-1}$
    else $r_i :=$ CONSTRUCT$(r_{i-1}, \varepsilon)$
          /* Call the function in $\mathcal{A}_{\mathcal{L}}$. */
    fi
    output $(r_i, X_i)$ as a conjecture for a
    language $\{\varepsilon\}$;
    $i := i + 1$;
    read the next positive example $w_i$
  od
  repeat (forever)
    $S_i := S_{i-1} \cup \{w_i\}$;  $\Sigma_i := \Sigma_{i-1} \cup \text{alph}(w_i)$;
    if $w_i \neq \varepsilon$ then $X_i :=$ UPDATE$(X_{i-1}, w_i)$
                /* Call the function in $\mathcal{A}_{\mathcal{X}}$. */
    else $X_i := X_{i-1}$   fi
    if $X_i \neq X_{i-1}$ then
      let a set $\Sigma'_i$ be given as $|\Sigma'_i| = |X_i|$,
      where $X_i \subset \Sigma^*_i$;
      let $\varphi_i$ be a bijection of $\Sigma'_i$ onto $X_i$;
      $W := \varphi_i^{-1}(S_i)$;
      $S'_0 := \emptyset$;  reset $r'_0$ so that $L(r'_0) = \emptyset$;
      $j := 1$;
      repeat
        $w'_j := \varphi_i^{-1}(w_j)$;  $W := W - \{w'_j\}$;
        $S'_j := S'_{j-1} \cup \{w'_j\}$;
        if $w'_j \in L(r'_{j-1})$ then $r'_j := r'_{j-1}$
        else $r'_j :=$ CONSTRUCT$(r'_{j-1}, w'_j)$
              /* Call the function in $\mathcal{A}_{\mathcal{L}}$. */
        fi
        $j := j + 1$
      until $W = \emptyset$;  /* When $i = j - 1$,
                 it holds that $S'_i = \varphi_i^{-1}(S_i)$. */
      $r_i := r'_i$
    else
      $\Sigma'_i := \Sigma'_{i-1}$;  $\varphi_i := \varphi_{i-1}$;
      $w'_i := \varphi_i^{-1}(w_i)$;
      $S'_i := S'_{i-1} \cup \{w'_i\}$;
      /* It holds that $S'_i = \varphi_i^{-1}(S_i)$. */
      if $w'_i \in L(r_{i-1})$ then $r_i := r_{i-1}$
      else $r_i :=$ CONSTRUCT$(r_{i-1}, w'_i)$
            /* Call the function in $\mathcal{A}_{\mathcal{L}}$. */
      fi
    fi
    output $(r_i, X_i)$ as a conjecture for a
    language $\varphi_i(L(r_i))$;
    $i := i + 1$;
    read the next positive example $w_i$
end
```

From Definition 6, for any language $L \in \mathcal{C}(\mathcal{L},\mathcal{X})$, we can show that $L = \varphi(L')$ for some $L' \in \mathcal{L}$ and some bijection φ of $\Sigma'_{L'}$ onto X, where $\Sigma'_{L'} = \text{alph}(L')$ $(\subseteq \Sigma')$ and $X \in \mathcal{X}$. The algorithm $\mathcal{A}_{\mathcal{C}(\mathcal{L},\mathcal{X})}$ outputs a sequence of pairs (r_i, X_i) $(i = 1, 2, \ldots)$ such that $L_i = \varphi_i(L(r_i)) \in \mathcal{C}(\mathcal{L},\mathcal{X})$, where $L_i \supseteq S_i = \{w_1, w_2, \ldots, w_i\}$, $\Sigma'_i = \text{alph}(\varphi_i^{-1}(S_i))$ and $X_i = \varphi_i(\Sigma'_i)$. Note that the algorithm $\mathcal{A}_{\mathcal{C}(\mathcal{L},\mathcal{X})}$ needs only a positive presentation $w_1, w_2, \ldots$ of a target language $L \in \mathcal{C}(\mathcal{L},\mathcal{X})$.

4.1 Correctness of the Identification Algorithm

In the case where the target language in $\mathcal{C}(\mathcal{L},\mathcal{X})$ is $\{\varepsilon\}$, the algorithm $\mathcal{A}_{\mathcal{C}(\mathcal{L},\mathcal{X})}$ outputs the conjecture (r, X) such that $L(r) = \{\varepsilon\}$ and $X = \emptyset$. Next we are concerned with a target language $L_* \in \mathcal{C}(\mathcal{L},\mathcal{X})$ such that $L_* \neq \emptyset$ and $L_* \neq \{\varepsilon\}$. From Definition 6, a target language L_* can be denoted by $L_* = \varphi_*(L'_*)$ for some $L'_* \in \mathcal{L}$ and some coding morphism $\varphi_* : \Sigma'^* \to \Sigma^*$ such that $X_* = \varphi_*(\Sigma_{L'_*})$ for some $X_* \in \mathcal{X}$ over Σ, where $\Sigma_{L'_*} = \text{alph}(L'_*)$ $(\subseteq \Sigma')$.

The next lemma comes from the algorithm $\mathcal{A}_{\mathcal{X}}$.

Lemma 3. Suppose that $\varphi_*(L'_*) \neq \emptyset$ and $\varphi_*(L'_*) \neq \{\varepsilon\}$. For any positive presentation of $\varphi_*(L'_*) - \{\varepsilon\}$, the algorithm $\mathcal{A}_{\mathcal{X}}$ identifies the coarsest code $X_{\varphi_*(L'_*)} \in \mathcal{X}$ which can factorize $\varphi_*(L'_*)$ in the limit.

Proof: Since $\varphi_*(L'_*) \neq \emptyset$ and $\varphi_*(L'_*) \neq \{\varepsilon\}$, $\varphi_*(L'_*) - \{\varepsilon\}$ is a nonempty set such that $\varphi_*(L'_*) - \{\varepsilon\} \subseteq X^+$ for some $X \in \mathcal{X}$. Then, from Lemma 2, there uniquely exists the coarsest code $X_{\varphi_*(L'_*)-\{\varepsilon\}} \in \mathcal{X}$ which can factorize $\varphi_*(L'_*) - \{\varepsilon\}$. Therefore, for any positive presentation of $\varphi_*(L'_*) - \{\varepsilon\}$, $\mathcal{A}_{\mathcal{X}}$ identifies $X_{\varphi_*(L'_*)-\{\varepsilon\}}$ in the limit. Thus, from Lemma 1, $X_{\varphi_*(L'_*)-\{\varepsilon\}}$ is identical to the coarsest code $X_{\varphi_*(L'_*)}$ which can factorize $\varphi_*(L'_*)$. □

Lemma 3 assures that there exists a large enough number N_1 such that, for each $i \geq N_1$, X_i in the algorithm $\mathcal{A}_{\mathcal{C}(\mathcal{L},\mathcal{X})}$ is identical to the coarsest code $X_{\varphi_*(L'_*)}$. Since Σ'_i and φ_i in $\mathcal{A}_{\mathcal{C}(\mathcal{L},\mathcal{X})}$ are not updated any more for $i > N_1$, we may let $\Sigma' = \Sigma'_i$ and $\varphi' = \varphi_i$. Then, the followings hold.

(1) From the above assumption, $X_* \in \mathcal{X}$ can factorize $\varphi_*(L'_*)$. For any $X' \in \mathcal{X}$ which can factorize $\varphi_*(L'_*)$, X' can factorize the coarsest code $X_{\varphi_*(L'_*)}$ from Definition 5. Therefore, X_* can factorize $X_{\varphi_*(L'_*)}$.

(2) φ' is a bijection of Σ' onto $X_{\varphi_*(L'_*)}$.

Furthermore, we may assume that $\Sigma' \subseteq \Sigma$. Then, the next key lemma holds.

Lemma 4. The language $\varphi'^{-1}(\varphi_*(L'_*))$ is in $\mathcal{L}$.

Proof: Since φ_* is a bijection of $\Sigma_{L'_*}$ onto X_* and X_* can factorize $X_{\varphi_*(L'_*)}$, it holds that $\varphi_*^{-1}(X_{\varphi_*(L'_*)}) \subseteq \varphi_*^{-1}((X_*)^*) = (\varphi_*^{-1}(X_*))^* = (\Sigma_{L'_*})^*$. Let $X' = \varphi_*^{-1}(X_{\varphi_*(L'_*)})$. Since $X_{\varphi_*(L'_*)}$ is the coarsest code which can factorize $\varphi_*(L'_*)$, it holds that X' is a code over $\Sigma_{L'_*}$ from Corollary 1 and that $L'_* = \varphi_*^{-1}(\varphi_*(L'_*)) \subseteq \varphi_*^{-1}((X_{\varphi_*(L'_*)})^*) = (\varphi_*^{-1}(X_{\varphi_*(L'_*)}))^* (= X'^*)$, i.e., X' can factorize L'_*.

In the algorithm $\mathcal{A}_{\mathcal{C}(\mathcal{L},\mathcal{X})}$, it holds that $\varphi'^{-1}(\varphi_*(L'_*)) \subseteq \varphi'^{-1}((X_{\varphi_*(L'_*)})^*) = (\varphi'^{-1}(X_{\varphi_*(L'_*)}))^* = \Sigma'^*$ since $\varphi_*(L'_*) \subseteq (X_{\varphi_*(L'_*)})^*$ and $X_{\varphi_*(L'_*)} = \varphi'(\Sigma')$.

Let $\psi = \varphi_*^{-1} \circ \varphi'$. Then, it holds that $\psi(\Sigma') = \varphi_*^{-1} \circ \varphi'(\Sigma') = \varphi_*^{-1}(\varphi'(\Sigma')) = \varphi_*^{-1}(X_{\varphi_*(L'_*)}) = X'$.

Since $L'_* \in \mathcal{L}$, X' is a code which can factorize L'_* (i.e., $L'_* \subseteq X'^*$), and ψ is a bijection of Σ' onto X' (i.e., $X' = \psi(\Sigma')$), it holds that $\psi^{-1}(L'_*) \in \mathcal{L}$ from Condition 1. Therefore, we have that $\varphi'^{-1}(\varphi_*(L'_*)) = \varphi'^{-1} \circ \varphi_*(L'_*) = (\varphi_*^{-1} \circ \varphi')^{-1}(L'_*) = \psi^{-1}(L'_*) \in \mathcal{L}$. □

A sequence $\varphi'^{-1}(w_1), \varphi'^{-1}(w_2), \ldots$ is a positive presentation of $\varphi'^{-1}(\varphi_*(L'_*))$ corresponding to a positive presentation $w_1, w_2, \ldots$ of $\varphi_*(L'_*)$. Since $\varphi'^{-1}(\varphi_*(L'_*)) \in \mathcal{L}$ from Lemma 4, the algorithm $\mathcal{A}_{\mathcal{L}}$ identifies $\varphi'^{-1}(\varphi_*(L'_*))$ in the limit from positive data. That is, there exists a large enough number N_2 such that, for each $i \geq N_2$, r_i is identical to $\tilde{r}$ such that $L(\tilde{r}) = \varphi'^{-1}(\varphi_*(L'_*))$. Then, for each time where $i \geq N_2$, $\mathcal{A}_{\mathcal{C}(\mathcal{L},\mathcal{X})}$ outputs a pair of $\tilde{r}$ and $X_{\varphi_*(L'_*)}$ such that $X_{\varphi_*(L'_*)} = \varphi'(\Sigma')$. Thus, we have that $\varphi'(L(\tilde{r})) = \varphi'(\varphi'^{-1}(\varphi_*(L'_*))) = \varphi_*(L'_*)$, where $\varphi_*(L'_*)$ is the target language. Then, we have the next theorem.

Theorem 1. The class $\mathcal{C}(\mathcal{L},\mathcal{X})$ of languages that satisfies Conditions 1 and 2 is identifiable in the limit from positive data. □

Note that Theorem 1 assures that there exists a large enough number N_2 such that, for each $i \geq N_2$, $\varphi_i(L(r_i)) = \varphi_*(L'_*)$, but it does not neccesarily hold that $L(r_i) = L'_*$, $\Sigma'_i = \Sigma_{L'_*} (= \mathrm{alph}(L'_*))$, $X_i = X_*$ and $\varphi' = \varphi_*$.

4.2 Time Analysis of the Identification Algorithm

Suppose that a sequence $w_1, w_2, \ldots$ is a positive presentation of the target language in $\mathcal{C}(\mathcal{L},\mathcal{X})$ that satisfies Conditions 1 and 2. Let $S_i = \{w_1, w_2, \ldots, w_i\}$ and $S'_i = \{\varphi'^{-1}(w_j) \mid w_j \in S_i - \{\varepsilon\}, 1 \leq j \leq i\}$ for each $i \geq 1$.

[Time for Updating a Conjecture] The time used by $\mathcal{A}_{\mathcal{C}(\mathcal{L},\mathcal{X})}$ between receiving the i-th example w_i and outputting the i-th conjecture (r_i, X_i) for a language $\varphi_i(L(r_i))$, where $X_i = \varphi_i(\Sigma'_i)$ and $\Sigma'_i = \mathrm{alph}(S'_i)$, is mainly given by the total time for computing the following three processes: (1) the function UPDATE(X_{i-1}, w_i), (2) the process computing the set S'_i, and (3) the functions CONSTRUCT(r'_{j-1}, S'_j) for all $1 \leq j \leq i$. The above processes (1) and (2) depend on the properties of the class $\mathcal{X}$, while the process (3) depends on the properties of the class $\mathcal{L}$. The time for computing the process (1) corresponds to the time used by $\mathcal{A}_{\mathcal{X}}$ for updating a conjecture X_i. The time for computing the process (2) is equal to the time for computing X_i-factorizations of w_j for all j $(1 \leq j \leq i)$. And then, the time for computing the process (3) corresponds to the total time used by $\mathcal{A}_{\mathcal{L}}$ between receiving the first example $\varphi_1^{-1}(w_1)$ and outputting the i-th conjecture r'_i.

[The Number of Implicit Errors] In the learning process of $\mathcal{A}_{\mathcal{C}(\mathcal{L},\mathcal{X})}$, whenever $X_i \neq X_{i-1}$, a set Σ'_i and a bijection φ_i are computed over again using by an updated code X_i. In this case, for each i, the number of updating conjectures $r'_j\,(1 \leq j \leq i)$ in $\mathcal{A}_{\mathcal{C}(\mathcal{L},\mathcal{X})}$ is bounded by the number of implicit errors of prediction made by $\mathcal{A}_{\mathcal{L}}$. Therefore, the number of implicit errors of prediction made by $\mathcal{A}_{\mathcal{C}(\mathcal{L},\mathcal{X})}$ is bounded by the number of implicit errors of prediction made by $\mathcal{A}_{\mathcal{L}}$ multiplied by that of prediction made by $\mathcal{A}_{\mathcal{X}}$.

5 Examples of Applications

We shall show some examples of applying this algorithm to pairs of some class of languages and that of codes in the followings.

Let $M = (Q, \Sigma, \delta, q_0, F)$ be a *deterministic finite automaton* (DFA, for short), where Q is the finite set of *states*, Σ is the finite set of *input symbols*, $\delta : Q \times \Sigma \to Q$ is the *transition function*, q_0 $(\in Q)$ is the *initial* state, and F $(\subseteq Q)$ is the set of *final* states. For any nonnegative integer k, a string $u \in \Sigma^*$ is said to be a *k-leader* of a state $q \in Q$ of M iff $|u| = k$ and there exists a state $p \in Q$ such that $\delta(p, u) = q$. At first, we define the following languages.

Definition 8. A DFA M is said to be a *restricted strictly deterministic automaton* (RSDA, for short) iff, for any $a \in \Sigma$, there exists at most one pair of states $(p, q) \in Q \times Q$ such that $\delta(p, a) = q$. The language accepted by an RSDA M is said to be a *restricted strictly regular language* (RSRL, for short). □

Definition 9. Let k be a positive integer. A DFA M is said to be a *k-definite* DFA (k-DDFA, for short) iff, for any pair of distinct states q_1 and q_2 in Q, there exists no string that is a k-leader of both q_1 and q_2. The language accepted by a k-DDFA M is said to be a *k-definite regular language* (k-DRL, for short). □

Let $\mathcal{RSR}$, $\mathcal{DR}_k$, and $\mathcal{R}ev_k$ be the class of RSRLs, the class of k-DRLs for each $k \geq 1$, and the class of *k-reversible* languages [2] for each $k \geq 0$, respectively. From Definitions 8 and 9, we can show that for each $k \geq 1$, the following relationships hold: $\mathcal{RSR} \subset \mathcal{DR}_k \subset \mathcal{R}ev_k$. Therefore, these classes are identifiable in the limit from positive data. Furthermore, we can prove the following lemma.

Lemma 5. All of the classes $\mathcal{RSR}$, $\mathcal{DR}_k$ for any $k \geq 1$, and $\mathcal{R}ev_k$ for any $k \geq 0$ are closed under inverse coding morphism.

Proof: We shall show that $\mathcal{RSR}$ is closed under inverse coding morphism.

For any $L \in \mathcal{RSR}$, there exists an RSDA $M = (Q, \Sigma, \delta, q_0, F)$ such that $L(M) = L$. For any code $X \subseteq \Sigma^*$ which can factorize L, it holds that $L \subseteq X^*$. Let Σ' be an alphabet such that $X = \varphi(\Sigma')$ for some coding morphism $\varphi : \Sigma'^* \to \Sigma^*$. Then, it holds that $\varphi^{-1}(L) \subseteq \varphi^{-1}(X^*) = (\varphi^{-1}(X))^* = \Sigma'^*$. Therefore, there exists a minimal DFA $M' = (Q', \Sigma', \delta', q'_0, F')$ such that $L(M') = \varphi^{-1}(L(M))$. Note that, for any $w \in \Sigma'^*$, it holds that $w \in L(M')$ iff $\varphi(w) \in L(M)$.

Suppose for the sake of contradiction that M' is not an RSDA. Then, for some $a \in \Sigma'$, M' has a pair of transition functions $\delta'(p_1, a) = q_1$ and $\delta'(p_2, a) = q_2$ such that $p_1 \neq p_2$ or $q_1 \neq q_2$. Since M' is minimal, it holds that $L(p_1) \neq L(p_2)$ or $L(q_1) \neq L(q_2)$. Let $u_1, u_2 \in \Sigma'^*$ be strings such that $\delta'(q'_0, u_1) = p_1$ and $\delta'(q'_0, u_2) = p_2$. And let $v_1, v_2 \in \Sigma'^*$ be strings such that $\delta'(q_1, v_1), \delta'(q_2, v_2) \in F'$. Then, it holds that $u_1 a v_1, u_2 a v_2 \in L(M')$. Therefore, in the RSDA M, it holds that $\varphi(u_1 a v_1), \varphi(u_2 a v_2) \in L(M)$. Then, there exist $r_1, r_2 \in Q$ such that $\delta(q_0, \varphi(u_1)) = r_1$ and $\delta(q_0, \varphi(u_2)) = r_2$. Furthermore, there exist $s_1, s_2 \in Q$ such that $\delta(r_1, \varphi(a)) = s_1$ and $\delta(r_2, \varphi(a)) = s_2$. Since $\varphi(a) \in X \subseteq \Sigma^+$ and M is an RSDA, it should hold that $r_1 = r_2$ and $s_1 = s_2$. Therefore, $\delta(q_0, \varphi(u_1)) = \delta(q_0, \varphi(u_2)) = r_1$ and $\delta(q_0, \varphi(u_1 a)) = \delta(q_0, \varphi(u_2 a)) = s_1$.

In the case where $p_1 \neq p_2$, by symmetry, we may assume that for some $z \in \Sigma'^*$, $z \in L(p_1)$ and $z \notin L(p_2)$. Then, it holds that $u_1 z \in L(M')$ and $u_2 z \notin L(M')$. Therefore, it holds that $\varphi(u_1 z) \in L(M)$ and $\varphi(u_2 z) \notin L(M)$. Since $\varphi(u_1 z) \in L(M)$ and $\delta(q_0, \varphi(u_1)) = \delta(q_0, \varphi(u_2))$, it holds that $\varphi(u_2 z) \in L(M)$. This is a contradiction.

In the case where $q_1 \neq q_2$, by symmetry, we may assume that for some $z \in \Sigma'^*$, $z \in L(q_1)$ and $z \notin L(q_2)$. Then, it holds that $u_1 a z \in L(M')$ and $u_2 a z \notin L(M')$. Therefore, it holds that $\varphi(u_1 a z) \in L(M)$ and $\varphi(u_2 a z) \notin L(M)$. Since $\varphi(u_1 a z) \in L(M)$ and $\delta(q_0, \varphi(u_1 a)) = \delta(q_0, \varphi(u_2 a))$, it holds that $\varphi(u_2 a z) \in L(M)$. This is a contradiction.

Therefore, M' is an RSDA. Thus, $\mathcal{RSR}$ is closed under inverse coding morphism.

In a similar way to this, we can prove that all the classes $\mathcal{DR}_k$ for any $k \geq 1$ and $\mathcal{R}ev_k$ for any

$k \geq 0$ are closed under inverse coding morphism. □

Thus, all of the classes $\mathcal{RSR}$, $\mathcal{DR}_k$ for any $k \geq 1$, and $\mathcal{R}ev_k$ for any $k \geq 0$ satisfy Condition 1 from Lemma 5.

For a string $w \in \Sigma^+$, firstchar(w) (respectively, lastchar(w)) denotes the first (last, resp.) symbol of w. Then, we define the following classes of codes.

Definition 10. Let X be a code over Σ. X is called a *strict prefix* code (respectively, a *strict suffix* code) iff, for any pair of distinct elements $x_1, x_2 \in X$, it holds that firstchar$(x_1) \neq$ firstchar(x_2) (lastchar$(x_1) \neq$ lastchar(x_2), resp.). □

Let $\mathcal{SP}$, $\mathcal{SS}$ be the class of strict prefix codes, and that of strict suffix codes, respectively.

We can show that the class $\mathcal{C}(\mathcal{RSR}, \mathcal{SP})$ of languages coincides with the class of strictly regular languages [9]. Also, since $\mathcal{RSR}$ coincides with the class of Szilard languages of linear grammars [7], the function CONSTRUCT(r'_{i-1}, w'_i) in $\mathcal{A}_{\mathcal{RSR}}$ can be written as almost the same procedure CONSTRUCT(Δ_i) in [9], p.166 except Σ'_i is used instead of T_i. When $\mathcal{A}_{\mathcal{RSR}}$ receives $w'_1, w'_2, \ldots, w'_i$ as input, the total time for updating conjectures of $\mathcal{A}_{\mathcal{RSR}}$ is bounded by $O(\cup_{j=1}^{i} |w'_j|)$. Furthermore, the number of implicit errors of prediction made by $\mathcal{A}_{\mathcal{RSR}}$ is bounded by $O(|\Sigma'|)$. In a similar way to this analysis, we can show that the total time for updating conjectures of $\mathcal{A}_{\mathcal{DR}_k}$ for each $k \geq 1$ is bounded by $O(\cup_{j=1}^{i} |w'_j|)$ and the number of implicit errors of prediction made by $\mathcal{A}_{\mathcal{DR}_k}$ is bounded by $O(|\Sigma'|^{k+1})$.

And then, the function UPDATE(X_{i-1}, w_i) in $\mathcal{A}_{\mathcal{SP}}$ is the same as the procedure UPDATE(T_{i-1}, w_i) in [9], p.164. In a similar way to [9], we can prove that this function outputs the coarsest code which can factorize $\{w_1, w_2, \ldots, w_i\}$. Therefore, the class $\mathcal{SP}$ satisfies Condition 2. Similarly, we can show that the class $\mathcal{SS}$ also satisfies Condition 2. When $\mathcal{A}_{\mathcal{SP}}$ (respectively, $\mathcal{A}_{\mathcal{SS}}$) receives $w_1, w_2, \ldots, w_i$ as input, the time used by $\mathcal{A}_{\mathcal{SP}}$ ($\mathcal{A}_{\mathcal{SS}}$, resp.) for updating conjectures is bounded by $O(|\Sigma| l^2)$, where $l = \text{Max}\{|w_1|, |w_2|, \ldots, |w_i|\}$. The number of implicit errors of prediction made by $\mathcal{A}_{\mathcal{SP}}$ ($\mathcal{A}_{\mathcal{SS}}$, resp.) is bounded by $O(|\Sigma| l)$.

Summarizing the above results, we have the next theorem.

Theorem 2. The class $\mathcal{C}(\mathcal{RSR}, \mathcal{SS})$ of languages, which is incomparable to the class $\mathcal{C}(\mathcal{RSR}, \mathcal{SP})$, is polynomial time identifiable in the limit from positive data. For each $k \geq 1$, the class $\mathcal{C}(\mathcal{DR}_k, \mathcal{SP})$ ($\mathcal{C}(\mathcal{DR}_k, \mathcal{SS})$, respectively) of languages is polynomial time identifiable in the limit from positive data when we regard k to be a constant. □

6 Conclusions

We have been concerned with a *unified* algorithm for extending classes of languages identifiable in the limit from positive data. When the extended class $\mathcal{C}(\mathcal{L}, \mathcal{X})$ of languages satisfies Conditions 1 and 2, $\mathcal{C}(\mathcal{L}, \mathcal{X})$ is identifiable in the limit from positive data. Then, we have presented an identification algorithm $\mathcal{A}_{\mathcal{C}(\mathcal{L}, \mathcal{X})}$ for the class $\mathcal{C}(\mathcal{L}, \mathcal{X})$ of languages in question.

References

[1] Angluin, D., *Inductive inference of formal languages from positive data*, Inform. and Control **45** (1980), 117-135.

[2] Angluin, D., *Inference of reversible languages*, J. ACM **29** (1982), 741-765.

[3] Berstel, J. and D. Perrin, "Theory of Codes", Academic Press, Inc., 1985.

[4] Gold, E. M., *Language identification in the limit*, Inform. and Control **10** (1967), 447-474.

[5] Harrison, M. A., "Introduction to Formal Language Theory", Addison-Wesley, Reading, Massachusetts, 1972.

[6] Kobayashi, S. and T. Yokomori, *Identifiability of subspaces and homomorphic images of zero-reversible languages*, ALT'97, LNAI **1316** (1997), 48-61.

[7] Mäkinen, E., *The grammatical inference problem for the Szilard languages of linear grammars*, Inform. Process. Lett. **36** (1990), 203-206.

[8] Pitt, L., *Inductive inference, DFAs, and computational complexity*, Proc. 2nd Workshop on Analogical and Inductive Inference, LNAI **397** (1989), 18-44.

[9] Yokomori, T., *On polynomial-time learnability in the limit of strictly deterministic automata*, Machine Learning **19** (1995), 153-179.