

A Catalog for Prediction-Preserving Reducibility with Membership Queries on Formal Languages

Kouichi Hirata¹, Hiroshi Sakamoto², and Hiroki Arimura²

¹ Department of Artificial Intelligence, Kyushu Institute of Technology
Kawazu 680-4, Iizuka 820-8502, Japan
hirata@ai.kyutech.ac.jp

² Department of Informatics, Kyushu University
Hakozaki 6-10-1, Fukuoka 812-8581, Japan
{hiroshi, arim}@i.kyushu-u.ac.jp

Abstract. This paper presents several results of *prediction-preserving reducibility with membership queries (pwm-reducibility)* on formal languages. We mainly deal with two kinds of concept classes, simple CFGs and finite unions of regular pattern languages. For the former, we show that DNF formulas are pwm-reducible to CFGs that is sequential or that contains at most one nonterminal. For the latter, on the other hand, we show that both bounded finite unions of regular pattern languages and unbounded finite unions of substring pattern languages are pwm-reducible to DFAs, while DNF formulas are pwm-reducible to unbounded finite unions of regular pattern languages.

1 Introduction

The task of predicting the classification of a new example is frequently discussed from the viewpoints of both *passive* and *active* settings. In a passive setting, the examples are all chosen independently according to a fixed but unknown probability distribution, and the learner has no control over selection of examples [12, 17]. In an active setting, on the other hand, the learner is allowed to ask about particular examples, that is, the learner makes *membership queries*, before the new example to predict is given to the learner [3, 6].

Concerned with language learning, we can design a polynomial-time algorithm to predict deterministic finite automata (DFAs) in an active setting [3], while predicting DFAs is as hard as computing certain apparently hard cryptographic predicates in a passive setting [12]. Furthermore, predicting nondeterministic finite automaton (NFAs) and unrestricted context-free grammars (CFGs) is also hard under the same cryptographic assumptions in an active setting [6]. Here, the *cryptographic assumptions* denote the intractability of inverting RSA encryption, recognizing quadratic residues and factoring Blum integers.

Pitt and Warmuth [17] have formalized the model of prediction and a reduction between two prediction problems that preserves polynomial-time

predictability called a *prediction-preserving reduction* in a passive setting. Angluin and Kharitonov [6] have extended to the prediction and the reduction in an active setting. The reduction is called a *prediction-preserving reduction with membership queries* or *pwm-reduction* for short. All of the above negative results rely on the prediction-preserving or pwm-reduction.

Note that the prediction is a weaker learning model than PAC-learning or query learning models; If a class is polynomial-time learnable with equivalence (and membership) queries, then it is polynomial-time PAC-learnable (with membership queries), and if a class is polynomial-time PAC-learnable (with membership queries), then it is polynomial-time predictable (with membership queries) [5, 6, 17].

However, the refined results of the pwm-reducibility except the above general ones, that is, the restricted CFGs or another languages such as pattern languages, have few found elsewhere. In particular, many reserchers have been interested in the pwm-reducibility on Boolean concepts, not on formal languages [6, 12, 17]. Hence, in this paper, we present the pwm-reducibility on simple CFGs and finite unions of regular pattern languages.

For the former, we introduce the following simple CFGs: *linear grammars* ($\mathcal{L}_{\text{linear}}$), *right-linear grammars* ($\mathcal{L}_{\text{right-linear}}$), and *left-linear grammars* ($\mathcal{L}_{\text{left-linear}}$) as usual; *k-bounded CFGs* [4] ($\mathcal{L}_{k\text{-bounded-CFG}}$) each of which right-hand side of productions contains at most k nonterminals; the *sequential CFGs* [7, 22] ($\mathcal{L}_{\text{sqCFG}}$) that the set of nonterminals has a partial order \leq such that $T \rightarrow vUw$ iff $T \leq U$ for nonterminals T and U ; the *properly sequential* or *loop-free CFGs* ($\mathcal{L}_{\text{psqCFG}}$) that is sequential but disallowed the occurrence of the same nonterminal in left- and right-hand sides in each production; the *k-CFGs* ($\mathcal{L}_{k\text{-CFG}}$) that contains at most k nonterminals; *parenthesis grammars* [18, 20] ($\mathcal{L}_{\text{paren}}$) that each production is of the form $T \rightarrow [w]$.

For the latter, we introduce the finite unions of regular pattern languages. A *pattern* is a string consisting of constant symbols and variables. A pattern is *regular* [23] if each variable in it occurs at most once. In particular, A regular pattern of the form xwy is called a *substring* pattern [24], where x and y are variables and w is a constant string. Furthermore, a *language of pattern* is the set of constant strings obtained by substituting nonempty constant strings for variables in the pattern. Then, we deal with the *bounded finite union* of regular pattern languages by some constant m ($\mathcal{L}_{\cup_m \text{RP}}$) and the *unbounded finite union* of regular or substring pattern languages ($\mathcal{L}_{\cup \text{RP}}$ or $\mathcal{L}_{\cup \text{subP}}$).

We denote that \mathcal{L}_1 is pwm-reducible to \mathcal{L}_2 with membership queries [6] by $\mathcal{L}_1 \trianglelefteq_{\text{pwm}} \mathcal{L}_2$. Furthermore, we denote that $\mathcal{L}_1 \trianglelefteq_{\text{pwm}} \mathcal{L}_2$ and $\mathcal{L}_2 \trianglelefteq_{\text{pwm}} \mathcal{L}_1$ by $\mathcal{L}_1 \cong_{\text{pwm}} \mathcal{L}_2$. Then, in this paper, we obtain the results described as Fig. 1. Hence, we obtain the following results on the polynomial-time predictability with membership queries.

1. $\mathcal{L}_{\text{linear}}$, $\mathcal{L}_{\text{right-linear}}$, $\mathcal{L}_{\text{left-linear}}$, $\mathcal{L}_{k\text{-bounded-CFG}}$ ($k \geq 1$), and $\mathcal{L}_{\text{paren}}$ are not polynomial-time predictable with membership queries under the cryptographic assumptions.

2. If $\mathcal{L}_{\text{sqCFG}}$, $\mathcal{L}_{\text{psqCFG}}$, and $\mathcal{L}_{k\text{-CFG}}$ are polynomial-time predictable with membership queries, then so are DNF formulas.
3. $\mathcal{L}_{\cup_m \text{RP}}$ ($m \geq 0$) and $\mathcal{L}_{\cup \text{subP}}$ are polynomial-time predictable with membership queries.
4. If $\mathcal{L}_{\cup \text{RP}}$ is polynomial-time predictable with membership queries, then so are DNF formulas.

$\mathcal{L}_{\text{NFA}} \cong_{\text{pwm}}$	$\mathcal{L}_{\text{right-linear}}, \mathcal{L}_{\text{left-linear}}$
$\mathcal{L}_{\text{NFA}} \trianglelefteq_{\text{pwm}}$	$\mathcal{L}_{\text{linear}}, \mathcal{L}_{k\text{-bounded-CFG}}$
$\mathcal{L}_{\text{DNF}} \trianglelefteq_{\text{pwm}}$	$\mathcal{L}_{\text{sqCFG}}, \mathcal{L}_{\text{psqCFG}}, \mathcal{L}_{k\text{-CFG}}$
$\mathcal{L}_{\cup \text{DFA}} \trianglelefteq_{\text{pwm}}$	$\mathcal{L}_{\text{paren}}$
	$\mathcal{L}_{\cup_m \text{RP}}, \mathcal{L}_{\cup \text{subP}} \trianglelefteq_{\text{pwm}} \mathcal{L}_{\text{DFA}}$
$\mathcal{L}_{\text{DNF}} \trianglelefteq_{\text{pwm}}$	$\mathcal{L}_{\cup \text{RP}}$

Fig. 1. The pwm-reducibility

2 Preliminaries

Let Σ and N be two non-empty finite sets of symbols such that $\Sigma \cap N = \emptyset$. A *production* $A \rightarrow \alpha$ on Σ and N is an association from a nonterminal $A \in N$ to a string $\alpha \in (N \cup \Sigma)^*$. A *context-free grammar* (CFG, for short) is a 4-tuple (N, Σ, P, S) , where $S \in N$ is the distinguished *start symbol* and P is a finite set of productions on Σ and N . Symbols in N are said to be *nonterminals*, while symbols in Σ *terminals*.

In this paper, we deal with the following subclasses of CFGs.

- A *linear grammar* is a CFG $G = (N, \Sigma, P, S)$ such that each production in P is of the forms $T \rightarrow wUv$ or $T \rightarrow w$ for $T, U \in N$ and $w, v \in \Sigma^*$. In particular, a *right-linear* (*resp.*, *left-linear*) *grammar* if it is a linear grammar such that each production is of the forms either $T \rightarrow wU$ (*resp.*, $T \rightarrow Uw$) or $T \rightarrow w$ for $T, U \in N$ and $w \in \Sigma^*$.
- A CFG $G = (N, \Sigma, P, S)$ is called *k-bounded* [4] if the right-hand side of each production in P has at most k nonterminals.
- A CFG $G = (N, \Sigma, P, S)$ is called *sequential* [7, 22] if the nonterminals in N are labeled $S = T_1, \dots, T_n$ such that, for each production $T_i \rightarrow w$, $w \in (\Sigma \cup \{T_j \mid i \leq j \leq n\})^*$. In particular, a sequential CFG satisfying that $w \in (\Sigma \cup \{T_j \mid i < j \leq n\})^*$ for each production $T_i \rightarrow w$ is called *properly sequential* or *loop-free*.
- A CFG $G = (N, \Sigma, P, S)$ is called a *k-CFG* if $|N| \leq k$.
- A *parenthesis grammar* [18, 20] is a CFG $G = (N, \Sigma \cup \{[,]\}, P, S)$ such that each production in P is of the form $T \rightarrow [w]$ for $T \in N$ and $w \in (N \cup \Sigma)^*$.

Let G be a CFG (N, Σ, S, P) and α and β be strings in $(\Sigma \cup N)^*$. We denote $\alpha \Rightarrow_G \beta$ if there exist $\alpha_1, \alpha_2 \in (\Sigma \cup N)^*$ such that $\alpha = \alpha_1 X \alpha_2$, $\beta = \alpha_1 \gamma \alpha_2$ and $X \rightarrow \gamma \in P$. We extend the relation \Rightarrow_G to the reflexive and transitive closure \Rightarrow_G^* . For a nonterminal $A \in N$, the *language* $L_G(A)$ of A is the set $\{w \in \Sigma^* \mid A \Rightarrow_G^* w\}$. The *language* $L(G)$ of G just refers to $L_G(S)$.

Next, we introduce the notions of *patterns* [2]. Let X be a countable set of *variables* such that $\Sigma \cap X = \emptyset$. A *pattern* is an element of $(\Sigma \cup X)^+$. A pattern π is called *regular* [23] if each variables in π occurs at most once. In particular, a regular pattern of the form xwy is called a *substring* pattern [24] for $x, y \in X$ and $w \in \Sigma^+$.

A *substitution* is a homomorphism from patterns to patterns that maps each symbol $a \in \Sigma$ to itself. A substitution that maps some variables to an empty string ε is called an ε -*substitution*. In this paper, we do not deal with ε -substitution. By $\pi\theta$, we denote the image of a pattern by a substitution θ . For a pattern π , the *pattern language* $L(\pi)$ is the set $\{w \in \Sigma^+ \mid w = \pi\theta \text{ for some substitution } \theta\}$.

3 Prediction with Membership Queries

Let U denote Σ^* . If w is a string, $|w|$ denotes its length. For each $n > 0$, $U^{[n]} = \{w \in U \mid |w| \leq n\}$. A *representation of concepts* \mathcal{L} is any subset of $U \times U$. We interpret an element $\langle u, w \rangle$ of $U \times U$ as consisting a *concept representation* u and an *example* w . The example w is a member of a concept u if $\langle u, w \rangle \in \mathcal{L}$. Define the *concept represented by* u as $\kappa_{\mathcal{L}}(u) = \{w \mid \langle u, w \rangle \in \mathcal{L}\}$. The *set of concepts represented by* \mathcal{L} is $\{\kappa_{\mathcal{L}}(u) \mid u \in U\}$.

To represent CFGs, we define the class \mathcal{L}_{CFG} as the set of pairs $\langle u, w \rangle$ such that u encodes a CFG G and $w \in L(G)$. Also we define the classes $\mathcal{L}_{\text{linear}}$, $\mathcal{L}_{\text{right-linear}}$, $\mathcal{L}_{\text{left-linear}}$, $\mathcal{L}_{k\text{-bounded-CFG}}$, $\mathcal{L}_{\text{seqCFG}}$, $\mathcal{L}_{\text{psqCFG}}$, $\mathcal{L}_{k\text{-CFG}}$, and $\mathcal{L}_{\text{paren}}$, corresponding to linear grammars, right-linear grammars, left-linear grammars, k -bounded CFGs, sequential CFGs, properly sequential CFGs, k -CFGs, and parenthesis grammars, respectively, as similar.

To represent finite unions of regular pattern languages, we define the class $\mathcal{L}_{\cup_m \text{RP}}$ as the set of pairs $\langle u, w \rangle$ such that u encodes m and a finite set π_1, \dots, π_m of m regular patterns and w is in the concept represented by c iff $w \in L(\pi_i)$ for at least one π_i . Similarly, we define the class $\mathcal{L}_{\cup \text{RP}}$ (*resp.*, $\mathcal{L}_{\cup \text{subP}}$) as the set of pairs $\langle u, w \rangle$ such that u encodes a finite set π_1, \dots, π_r of regular (*resp.*, substring) patterns and w is in the concept represented by c iff $w \in L(\pi_i)$ for at least one π_i . Note that $\mathcal{L}_{\cup_m \text{RP}}$ denotes the *bounded* finite unions, whereas $\mathcal{L}_{\cup \text{RP}}$ and $\mathcal{L}_{\cup \text{subP}}$ denote the *unbounded* finite unions.

Additionally, we introduce the following classes. The class \mathcal{L}_{DFA} (*resp.*, \mathcal{L}_{NFA}) denotes the set of pairs $\langle u, w \rangle$ such that u encodes a DFA (*resp.*, NFA) M and M accepts w . The class $\mathcal{L}_{\cup \text{DFA}}$ of finite union of DFAs denotes the set of pairs $\langle u, w \rangle$ such that u encodes a finite set M_1, \dots, M_r of DFAs and w is in the concept represented by c iff at least one M_i accepts w . The class \mathcal{L}_{DNF} denotes the set of pairs $\langle u, w \rangle$ such that u encodes a positive integer n and a DNF formula d

over n Boolean variables x_1, \dots, x_n such that $|w| = n$ ($w = w_1 \cdots w_n$) and the assignment $x_i = w_i$ ($1 \leq i \leq n$) satisfies d .

Angluin and Kharitonov [6] have generalized the definitions of Pitt and Warmuth of prediction algorithm [17] to allow membership queries as follows.

Definition 1 (Angluin & Kharitonov [6]). A *prediction with membership queries algorithm*, or *pwm-algorithm*, is a possibly randomized algorithm A that takes as input n (a bound on the size of examples), s (a bound on the size of the target concept representations), and ε (an accuracy bound). It may make three different kinds of oracle calls, the responses to which are determined by the unknown target concept c_* and the unknown distribution D on $U^{[n]}$.

1. A *membership query* [3, 6] takes a string $w \in U$ as input and returns 1 if $w \in c_*$; and 0 otherwise.
2. A *request for random classified example* takes no input and returns a pair $\langle w, b \rangle$, where w is a string chosen independently according to D and $b = 1$ if $w \in c_*$ and $b = 0$ otherwise.
3. A *request for an element to predict* takes no input and returns a string w chosen independently according to D .

A may make any number of membership queries or requests for random classified examples, whereas A must eventually make one and only one request for an element to predict and then eventually halt with an output 0 or 1 without making any further oracle calls. The output is interpreted as A 's guess of how the target concept classifies the element returned by the request for an element to predict. A *runs in polynomial time* if its running time (counting one step per oracle call) is bounded by a polynomial in n , s and $1/\varepsilon$.

Definition 2 (Angluin & Kharitonov [6]). Let \mathcal{L} be a representation of concepts and c_* be the unknown target concept in \mathcal{L} . We say that A *successfully predicts* \mathcal{L} if, for each positive integer n and s , for each positive rational ε , for each concept representation $u \in U^{[n]}$, for each probability distribution D on $U^{[n]}$, when A is run with input n , s and ε , and oracles determined by $c_* = \kappa_{\mathcal{L}}(u)$ and D , A asks membership queries that are in U and the probability in at most ε that the output of A is not equal to the correct classification of w by $\kappa_{\mathcal{L}}(u)$, where w is the string returned by the (unique) request for an element of predict.

Definition 3 (Angluin & Kharitonov [6]). A representation \mathcal{L} of concepts is *polynomial-time predictable with membership queries* if there exists a pwm-algorithm A that runs in polynomial time and successfully predicts \mathcal{L} .

It is well known the following statements:

1. \mathcal{L}_{DFA} is polynomial-time predictable with membership queries [3].
2. $\mathcal{L}_{\text{UDFA}}$, \mathcal{L}_{NFA} and \mathcal{L}_{CFG} are not polynomial-time predictable with membership queries under the cryptographic assumptions [6].
3. \mathcal{L}_{DNF} is either polynomial-time predictable or not polynomial-time predictable with membership queries, if there exist one-way functions that cannot be inverted by polynomial-sized circuits [6].

Prediction-preserving reducibility introduced by Pitt and Warmuth [17] is a tool for showing that one class of representations is easier or harder to predict than another. Angluin and Kharitonov [6] have extended it to the prediction-preserving reduction *with membership queries*.

Definition 4 (Angluin & Kharitonov [6]). Let \mathcal{L}_i be a representation of concepts over domain U_i ($i = 1, 2$). We say that *predicting \mathcal{L}_1 reduces to predicting \mathcal{L}_2 with membership queries* (*pwm-reduces*, for short), denoted by $\mathcal{L}_1 \trianglelefteq_{\text{pwm}} \mathcal{L}_2$, if there exist an *instance mapping* $f : \mathbf{N} \times \mathbf{N} \times U_1 \rightarrow U_2$, a *concept mapping* $g : \mathbf{N} \times \mathbf{N} \times \mathcal{L}_1 \rightarrow \mathcal{L}_2$, and a *query mapping* $h : \mathbf{N} \times \mathbf{N} \times U_2 \rightarrow U_1 \cup \{\top, \perp\}$ satisfying the following conditions.

1. For each $x \in U_1^{[n]}$ and $u \in \mathcal{L}_1^{[s]}$, $x \in \kappa_{\mathcal{L}_1}(u)$ iff $f(n, s, x) \in \kappa_{\mathcal{L}_2}(g(n, s, u))$.
2. f is computable in time bounded by a polynomial in n , s and $|x|$.
3. The size of $g(n, s, u)$ is bounded by a polynomial in n , s and $|u|$.
4. For each $x' \in U_2$ and $u \in \mathcal{L}_1^{[s]}$, if $h(n, s, x') = \top$ then $x' \in \kappa_{\mathcal{L}_2}(g(n, s, u))$; if $h(n, s, x') = \perp$ then $x' \notin \kappa_{\mathcal{L}_2}(g(n, s, u))$; if $h(n, s, x') = x \in U_1$, then it holds that $x' \in \kappa_{\mathcal{L}_2}(g(n, s, u))$ iff $x \in \kappa_{\mathcal{L}_1}(u)$.
5. h is computable in time bounded by a polynomial in n , s and $|x'|$.

If $\mathcal{L}_1 \trianglelefteq_{\text{pwm}} \mathcal{L}_2$ and $\mathcal{L}_2 \trianglelefteq_{\text{pwm}} \mathcal{L}_1$, we denote $\mathcal{L}_1 \cong_{\text{pwm}} \mathcal{L}_2$.

The following theorem is useful for showing the predictability or the hardness of predictability of the class of representations.

Theorem 1 (Angluin & Kharitonov [6]). Let \mathcal{L}_1 and \mathcal{L}_2 be representations of concepts and suppose that $\mathcal{L}_1 \trianglelefteq_{\text{pwm}} \mathcal{L}_2$.

1. If \mathcal{L}_2 is polynomial-time predictable with membership queries, then so is \mathcal{L}_1 .
2. If \mathcal{L}_1 is not polynomial-time predictable with membership queries, then neither is \mathcal{L}_2 .

4 Prediction-Preserving Reducibility with Membership Queries

In this section, we fix f , g and h to an instance mapping, a concept mapping, and a query mapping. Furthermore, the parameters n and s denote the bounds of examples and representations, respectively. For simplicity, we assume that the length of examples of Boolean concepts is always fixed to the upper bound n .

4.1 Simple CFGs

First of all, by using the transformation from a DFA to a right-linear grammar (cf. [8, 9]), it holds that $\mathcal{L}_{\text{DFA}} \trianglelefteq_{\text{pwm}} \mathcal{L}_{\text{right-linear}}$, because the size of the right-linear grammar is bounded by a polynomial in the size of a DFA. Note that the converse direction $\mathcal{L}_{\text{right-linear}} \trianglelefteq_{\text{pwm}} \mathcal{L}_{\text{DFA}}$ does not follow from the transformation from a right-linear grammar to a DFA, because the size of the DFA is not bounded by a polynomial in the size of a right-linear grammar in general.

Furthermore, we point out that $\mathcal{L}_{\text{DNF}} \trianglelefteq_{\text{pwm}} \mathcal{L}_{\text{UDFA}}$, while $\mathcal{L}_{\text{DNF}} \trianglelefteq \mathcal{L}_{\text{DFA}}$ [17]. Here, \trianglelefteq means the prediction-preserving reduction without membership queries introduced by Pitt and Warmuth [17], that is, there exists an instance mapping and a concept mapping satisfying the requirement from 1 to 3 in Definition 4. Note that we cannot apply the same proof of $\mathcal{L}_{\text{DNF}} \trianglelefteq \mathcal{L}_{\text{DFA}}$ [17] to proving $\mathcal{L}_{\text{DNF}} \trianglelefteq_{\text{pwm}} \mathcal{L}_{\text{DFA}}$; We cannot construct a query mapping h .

On the other hand, by regarding the equivalent transformation between a NFA and a right-linear grammar [8, 9] as a concept mapping g , we observe that $\mathcal{L}_{\text{NFA}} \cong_{\text{pwm}} \mathcal{L}_{\text{right-linear}}$. Furthermore, for a CFG $G = (N, \Sigma, P, S)$, let G^R be a CFG (N, Σ, P', S) such that $T \rightarrow w^R \in P'$ for each $T \rightarrow w \in P$. Here, R denotes the reversal of a word. Then, for a right-linear (*resp.*, left-linear) grammar G , construct the following f , g and h :

$$\begin{aligned} f(n, s, e) &= e^R, \\ g(n, s, G) &= G^R, \\ h(n, s, e') &= e'^R. \end{aligned}$$

It is obvious that $\mathcal{L}_{\text{right-linear}} \trianglelefteq_{\text{pwm}} \mathcal{L}_{\text{left-linear}}$ (*resp.*, $\mathcal{L}_{\text{left-linear}} \trianglelefteq_{\text{pwm}} \mathcal{L}_{\text{right-linear}}$), so it holds that $\mathcal{L}_{\text{right-linear}} \cong_{\text{pwm}} \mathcal{L}_{\text{left-linear}}$. Furthermore, we also observe that $\mathcal{L}_{\text{NFA}} \trianglelefteq_{\text{pwm}} \mathcal{L}_{\text{linear}}$ and $\mathcal{L}_{\text{NFA}} \trianglelefteq_{\text{pwm}} \mathcal{L}_{k\text{-bounded-CFG}}$ for each $k \geq 1$. Summary:

Theorem 2. $\mathcal{L}_{\text{NFA}} \cong_{\text{pwm}} \mathcal{L}$ for $\mathcal{L} \in \{\mathcal{L}_{\text{right-linear}}, \mathcal{L}_{\text{left-linear}}\}$. Also, $\mathcal{L}_{\text{NFA}} \trianglelefteq_{\text{pwm}} \mathcal{L}_{\text{linear}}$ and $\mathcal{L}_{\text{NFA}} \trianglelefteq_{\text{pwm}} \mathcal{L}_{k\text{-bounded-CFG}}$ for each $k \geq 1$.

Theorem 3. $\mathcal{L}_{\text{DNF}} \trianglelefteq_{\text{pwm}} \mathcal{L}$ for $\mathcal{L} \in \{\mathcal{L}_{\text{psqCFG}}, \mathcal{L}_{\text{sqCFG}}\}$.

Proof. Let d be a DNF formula $t_1 \vee \dots \vee t_m$ over n Boolean variables x_1, \dots, x_n . First, we define w_i^j ($1 \leq i \leq n, 1 \leq j \leq m$) as follows:

$$w_i^j = \begin{cases} 1 & \text{if } t_j \text{ contains } x_i, \\ 0 & \text{if } t_j \text{ contains } \bar{x}_i, \\ T & \text{otherwise.} \end{cases}$$

Then, construct f , g and h as follows:

$$\begin{aligned} f(n, s, e) &= e, \\ g(n, s, d) &= (\{S, T\}, \{0, 1\}, S, \{S \rightarrow w_1^1 \dots w_n^1 \mid \dots \mid w_1^m \dots w_n^m, T \rightarrow 0 \mid 1\}), \\ h(n, s, e') &= e'. \end{aligned}$$

It is obvious that the above f , g and h satisfy the conditions of Definition 4. \square

Theorem 4. For each $k \geq 1$, $\mathcal{L}_{\text{DNF}} \trianglelefteq_{\text{pwm}} \mathcal{L}_{k\text{-CFG}}$.

Proof. Theorem 3 implies that $\mathcal{L}_{\text{DNF}} \trianglelefteq_{\text{pwm}} \mathcal{L}_{k\text{-CFG}}$ for each $k \geq 2$, so it is sufficient to show that $\mathcal{L}_{\text{DNF}} \trianglelefteq_{\text{pwm}} \mathcal{L}_{1\text{-CFG}}$. Let $d = t_1 \vee \dots \vee t_m$ be a DNF formula over n Boolean variables x_1, \dots, x_n . First, define w_i^j ($1 \leq i \leq n, 1 \leq j \leq m$) as follows:

$$w_i^j = \begin{cases} 1 & \text{if } t_j \text{ contains } x_i, \\ 0 & \text{if } t_j \text{ contains } \bar{x}_i, \\ S & \text{otherwise.} \end{cases}$$

Then, construct f , g and h as follows:

$$\begin{aligned}
f(n, s, e) &= e, \\
g(n, s, d) &= (\{S\}, \{0, 1\}, S, \\
&\quad \{S \rightarrow 0 \mid 1 \mid w_1^1 \cdots w_n^1 \mid \cdots \mid w_1^m \cdots w_n^m \mid \underbrace{S \cdots S}_{n+1} \mid \cdots \mid \underbrace{S \cdots S}_{2n}\}), \\
h(n, s, e') &= \begin{cases} e' & \text{if } |e'| = n, \\ \perp & \text{if } 1 < |e'| < n, \\ \top & \text{if } |e'| = 1 \text{ or } |e'| > n. \end{cases}
\end{aligned}$$

For each $e \in \{0, 1\}^n$, it holds that e satisfies d iff $S \Rightarrow_{g(n, s, d)}^* f(n, s, e)$. Furthermore, for each $e' \in \{0, 1\}^*$, if $h(n, s, e') = \perp$, then $S \not\Rightarrow_{g(n, s, d)}^* e'$, because $g(n, s, d)$ generates no strings of length more than 1 and less than n ; If $h(n, s, e') = e'$, then it holds that $S \Rightarrow_{g(n, s, d)}^* e'$ iff $h(n, s, e')$ satisfies d .

Finally, consider the case that $h(n, s, e') = \top$. It is sufficient to show that, for each $k \geq 1$, it holds that $S \Rightarrow_{g(n, s, d)}^* \underbrace{S \cdots S}_{kn+m}$ for each m ($1 \leq m \leq n$).

If $k = 1$, then, by the definition, it holds that $S \Rightarrow_{g(n, s, d)}^* \underbrace{S \cdots S}_{n+m}$ for each m ($1 \leq m \leq n$). Suppose that it holds that, for some $k \geq 1$, $S \Rightarrow_{g(n, s, d)}^* \underbrace{S \cdots S}_{kn+m}$

for each m ($1 \leq m \leq n$). Then, it holds that $S \Rightarrow_{g(n, s, d)}^* \underbrace{S \cdots S}_{kn+(m-1)} S \Rightarrow_{g(n, s, d)}^* \underbrace{S \cdots S}_{kn+(m-1)}$

$\underbrace{S \cdots S}_{kn+(m-1)} \underbrace{S \cdots S}_{n+1} = \underbrace{S \cdots S}_{(k+1)n+m}$ for each m ($1 \leq m \leq n$). Hence, $g(n, s, d)$ generates all strings of length more than n , so if $h(n, s, e') = \top$, then $S \Rightarrow_{g(n, s, d)}^* e'$. \square

Theorem 5. $\mathcal{L}_{\cup\text{DFA}} \trianglelefteq_{\text{pwm}} \mathcal{L}_{\text{paren}}$.

Proof. Let M_1, \dots, M_r be DFAs with the same alphabet Σ ($[,] \notin \Sigma$) and with mutually distinct states. For each $M_i = (Q_i, \Sigma, \delta_i, q_0^i, F_i)$ ($1 \leq i \leq r$), construct a parenthesis grammar $G_i(n, s, M_i) = (Q_i, \Sigma \cup \{[,]\}, P_i, q_0^i)$ such that $q \rightarrow [a\delta_i(q, a)] \in P_i$ for each $q \in Q_i$ and $a \in \Sigma$; $q \rightarrow [\varepsilon] \in P_i$ for each $q \in F_i$, where ε is an empty string. By using $G_i(n, s, M_i)$, let P_{M_1, \dots, M_r} be the following set of productions for $S \notin (\cup_{1 \leq i \leq r} Q_i) \cup \Sigma \cup \{[,]\}$:

$$P_{M_1, \dots, M_r} = \{S \rightarrow [q_0^1] \mid \cdots \mid [q_0^r]\} \cup (\cup_{1 \leq i \leq r} P_i).$$

Then, construct f , g and h as follows:

$$\begin{aligned}
f(n, s, e_1 e_2 \cdots e_l) &= [[e_1[e_2[\cdots[e_l[\varepsilon]]\cdots]]]] \text{ for } e_i \in \Sigma, \\
g(n, s, \{M_1, \dots, M_r\}) &= (\{S\} \cup (\cup_{1 \leq i \leq r} Q_i), \Sigma \cup \{[,]\}, S, P_{M_1, \dots, M_r}), \\
h(n, s, e') &= \begin{cases} e_1 \cdots e_l & \text{if } e' = [[e_1[e_2[\cdots[e_l[\varepsilon]]\cdots]]]] \text{ and } e_i \in \Sigma, \\ \perp & \text{otherwise.} \end{cases}
\end{aligned}$$

Note that $L(g(n, s, \{M_1, \dots, M_r\})) \subseteq \{[[e_1[e_2[\cdots[e_m[\varepsilon]]\cdots]]]] \mid m \geq 1, e_i \in \Sigma\}$, so if $h(n, s, e') = \perp$, then $S \not\Rightarrow_{g(n, s, \{M_1, \dots, M_r\})}^* e'$. Also it is obvious that $S \Rightarrow_{g(n, s, \{M_1, \dots, M_r\})}^* [[e_1[e_2[\cdots[e_l[\varepsilon]]\cdots]]]]$ iff $e_1 e_2 \cdots e_l \in L(M_i)$ for some i ($1 \leq i \leq r$). Hence, it holds that $\mathcal{L}_{\cup\text{DFA}} \trianglelefteq_{\text{pwm}} \mathcal{L}_{\text{paren}}$. \square

Sakakibara [18] has shown that $\mathcal{L}_{\text{paren}}$ is polynomial-time predictable with membership and equivalence queries if the structural information is available. Furthermore, Sakamoto [20] has shown that $\mathcal{L}_{\text{paren}}$ is polynomial-time predictable with membership queries and *characteristic examples*. The above theorem claims that the structural information or the characteristic examples are essential for efficient learning of $\mathcal{L}_{\text{paren}}$.

4.2 Finite unions of regular pattern languages

In this section, we present the prediction-preserving reducibility with membership queries on bounded or unbounded finite unions of regular pattern languages.

Since each regular pattern language is regular [23], we can construct a DFA M_π such that $L(M_\pi) = L(\pi)$ for each regular pattern π as follows: Suppose that π is a regular pattern of the form $\pi = x_0\alpha_1x_1\alpha_2\cdots x_{n-1}\alpha_nx_n$, where $x_i \in X$ and $\alpha_i = a_1^i a_2^i \cdots a_{m_i}^i \in \Sigma^+$. Then, the *corresponding DFA* M_π of π is a DFA $(\Sigma, Q, \delta, q_0, F)$ such that:

1. $Q = \{q_0, p_1^1, \dots, p_{m_1}^1, q_1, p_1^2, \dots, p_{m_2}^2, q_2, \dots, q_{n-1}, p_1^n, \dots, p_{m_n}^n, q_n\}$ and $F = \{q_n\}$,
2. $\delta(q_i, a) = p_1^{i+1}$ and $\delta(q_n, a) = q_n$ for each $a \in \Sigma$ and $0 \leq i \leq n-1$,
3. $\delta(p_j^i, a_j^i) = p_{j+1}^i$ and $\delta(p_{m_i}^i, a_{m_i}^i) = q_i$ for each $1 \leq i \leq n$ and $1 \leq j \leq m_i - 1$,
4. $\delta(p_j^i, a) = p_1^i$ for each $a \in \Sigma$ such that $a \neq a_j^i$.

It is obvious that $|M_\pi|$ is bounded by a polynomial in $|\pi|$.

By using the corresponding DFAs, we can easily show that $\mathcal{L}_{\text{RP}} \leq_{\text{pwm}} \mathcal{L}_{\text{DFA}}$ by constructing the following f , g and h for each regular pattern π :

$$\begin{aligned} f(n, s, e) &= e, \\ g(n, s, \pi) &= M_\pi, \\ h(n, s, e') &= e'. \end{aligned}$$

Then, \mathcal{L}_{RP} is polynomial-time predictable with membership queries, which is implied by the result of Matsumoto and Shinohara [15] that \mathcal{L}_{RP} is polynomial-time learnable with equivalence and membership queries. Furthermore, the following theorem holds:

Theorem 6. *For each $m \geq 0$, $\mathcal{L}_{\cup_m \text{RP}} \leq_{\text{pwm}} \mathcal{L}_{\text{DFA}}$.*

Proof. Let π_1, \dots, π_m be m regular patterns. Also let $M_{\pi_i} = (Q_i, \Sigma, \delta_i, q_0^i, F_i)$ be the corresponding DFA of π_i . First, construct a DFA $M_{\pi_1, \dots, \pi_m} = (Q_1 \times \cdots \times Q_m, \Sigma, \delta, (q_0^1, \dots, q_0^m), F_1 \times \cdots \times F_m)$ such that $\delta((q_1, \dots, q_m), a) = (p_1, \dots, p_m)$ iff $\delta_i(q_i, a) = p_i$ for each i ($1 \leq i \leq m$). Then, construct f , g and h as follows:

$$\begin{aligned} f(n, s, e) &= e, \\ g(n, s, \{\pi_1, \dots, \pi_m\}) &= M_{\pi_1, \dots, \pi_m}, \\ h(n, s, e') &= e'. \end{aligned}$$

Note that the size of $g(n, s, \{\pi_1, \dots, \pi_m\})$ is bounded by a polynomial in s , i.e., $O(s^m)$. It is obvious that $L(\pi_1) \cup \cdots \cup L(\pi_m) = L(M_{\pi_1, \dots, \pi_m})$, which implies that $\mathcal{L}_{\cup_m \text{RP}} \leq_{\text{pwm}} \mathcal{L}_{\text{DFA}}$. \square

Theorem 7. $\mathcal{L}_{\text{DNF}} \leq_{\text{pwm}} \mathcal{L}_{\text{URP}}$.

Proof. Let $d = t_1 \vee \dots \vee t_m$ be a DNF formula over n Boolean variables x_1, \dots, x_n . First, for each term t_j ($1 \leq j \leq m$), construct a regular pattern $\pi_j = \pi_1^j \dots \pi_n^j$ as follows:

$$\pi_i^j = \begin{cases} 1 & \text{if } t_j \text{ contains } x_i, \\ 0 & \text{if } t_j \text{ contains } \overline{x_i}, \\ x_i^j & \text{otherwise.} \end{cases}$$

Furthermore, let π be a regular pattern $x_1 \dots x_n x_{n+1}$. Then, construct f , g and h as follows:

$$\begin{aligned} f(n, s, e) &= e, \\ g(n, s, d) &= \{\pi_1, \dots, \pi_m, \pi\}, \\ h(n, s, e') &= \begin{cases} e' & \text{if } |e'| = n, \\ \top & \text{if } |e'| > n, \\ \perp & \text{if } |e'| < n. \end{cases} \end{aligned}$$

For each $e' \in \{0, 1\}^*$, we can check the properties of h in Definition 4 as follows. Since $L(\pi) = \{w \in \{0, 1\}^* \mid |w| \geq n + 1\}$, if $h(n, s, e') = \top$, then $e' \in \kappa_{\mathcal{L}_{\text{URP}}}(g(n, s, d))$. On the other hand, since $|\pi_j| = n$ ($1 \leq j \leq m$) and $|\pi| = n + 1$, $\kappa_{\mathcal{L}_{\text{URP}}}(g(n, s, d))$ contains no strings of length $< n$. So, if $h(n, s, e') = \perp$, then $e' \notin \kappa_{\mathcal{L}_{\text{URP}}}(g(n, s, d))$. Otherwise, i.e. if $h(n, s, e') = e'$, note that $|e'| = n$, so $e' \notin L(\pi)$. Then, $e' \in L(\pi_1) \cup \dots \cup L(\pi_m)$. Thus, there exists an index i ($1 \leq i \leq m$) such that $e' \in L(\pi_i)$ iff e' is obtained by replacing the variables in π_i with 0 or 1, which is corresponding to a truth assignment satisfying t_i . Hence, $e' \in \kappa_{\mathcal{L}_{\text{URP}}}(g(n, s, d))$ iff $e' \in \kappa_{\mathcal{L}_{\text{DNF}}}(d)$.

Furthermore, for each $e \in \{0, 1\}^n$, $e \in \kappa_{\mathcal{L}_{\text{DNF}}}(d)$ iff $f(n, s, e) \in \kappa_{\mathcal{L}_{\text{URP}}}(g(n, s, d))$. Hence, it holds that $\mathcal{L}_{\text{DNF}} \leq_{\text{pwm}} \mathcal{L}_{\text{URP}}$. \square

Theorem 8. $\mathcal{L}_{\text{SubP}} \leq_{\text{pwm}} \mathcal{L}_{\text{DFA}}$.

Proof. Let π_1, \dots, π_r be substring patterns such that $\pi_i = x_i w_i y_i$. For a set $\{w_1, \dots, w_r\}$ of constant strings, consider the following modification of a *pattern matching machine (pmm)* M_{w_1, \dots, w_r} [1]. The goto function is defined as same as a pmm. The right-most constant in a string $w \in \Sigma^+$ is called a *last* of w . Then, the failure function *failure* for a non-last of each string w_i is defined as same as a pmm; For a last of w_i indexed by j , $\text{failure}(j) = j$. The output function is not necessary.

Since this modified pmm is also a DFA, so construct f , g and h as follows:

$$\begin{aligned} f(n, s, e) &= e, \\ g(n, s, \{\pi_1, \dots, \pi_r\}) &= M_{w_1, \dots, w_r}, \\ h(n, s, e') &= e'. \end{aligned}$$

Note that the size of $g(n, s, \{\pi_1, \dots, \pi_r\})$ is $O(|w_1| + \dots + |w_r|)$ [1]. Furthermore, it is obvious that $L(g(n, s, \{\pi_1, \dots, \pi_r\})) = L(\pi_1) \cup \dots \cup L(\pi_r)$, which implies that $\mathcal{L}_{\text{SubP}} \leq_{\text{pwm}} \mathcal{L}_{\text{DFA}}$. \square

Shinohara and Arimura [24] have discussed the inferability of \mathcal{L}_{U_mRP} , \mathcal{L}_{URP} and \mathcal{L}_{UsubP} in the framework of inductive inference. They have shown that \mathcal{L}_{U_mRP} and \mathcal{L}_{UsubP} are inferable from positive data, whereas \mathcal{L}_{URP} is not. In contrast, by Theorem 6, 7 and 8, \mathcal{L}_{U_mRP} and \mathcal{L}_{UsubP} are polynomial-time predictable with membership queries, whereas \mathcal{L}_{URP} is not polynomial-time predictable with membership queries if neither are DNF formulas.

5 Conclusion

In this paper, we have presented the pwm-reducibility on formal languages and obtained the results described as Fig. 1 in Section 1.

The results in Section 4.1 tell us that the efficient predictability of CFGs may be necessary to assume the deterministic concept. Ishizaka [10] has shown that *simple deterministic grammars* is polynomial-time learnable with *extended* equivalence and membership queries. The extended equivalence query can check the hypothesis not generated by simple deterministic grammars. It is open whether simple deterministic grammars is polynomial-time predictable with membership queries.

Angluin [4] has shown that $\mathcal{L}_{k\text{-bounded-CFG}}$ is polynomial-time predictable with *nonterminal* membership queries, and Sakakibara [19] has extended Angluin's result to *extended simple formal systems*. Although we have already tried to extend the pwm-reduction to prediction-preserving reduction with nonterminal membership queries partially [21], it is necessary to formulate and investigate in more detail.

In Section 4.2, we only deal with finite unions of *regular* pattern languages. Many researchers have developed the learnability/predictability of non-regular pattern languages such as [2, 11, 13–16, 24]. In particular, the learnability of the languages of *k-variable patterns* [2, 11] that contain at most k variables, *k μ -pattern* [15] each of which variable occurs at most k -times, and *erasing patterns* [16, 24] that allow to empty substitutions have been widely studied in the various learning frameworks. It is a future work to investigate the pwm-reducibility of them or their finite unions.

References

1. A. V. Aho, M. J. Corasick, *Efficient string matching: An aid to bibliographic search*, Comm. ACM **18**(6) (1975), 333–340.
2. D. Angluin, *Finding patterns common to a set of strings*, J. Comput. System Sci. **21** (1980) 46–62.
3. D. Angluin, *Learning regular sets from queries and counterexamples*, Inform. Comput. **75** (1987) 87–106.
4. D. Angluin, *Learning k-bounded context-free grammars*, Technical Report YALEU/DCS/RR-557, Yale University, 1987.
5. D. Angluin, *Queries and concept learning*, Mach. Learn. **2** (1988) 319–342.
6. D. Angluin, M. Kharitonov, *When won't membership queries help?*, J. Comput. System Sci. **50** (1995) 336–355.

7. A. Ginsburg, *The mathematical theory of context free languages* (McGraw-Hill, 1966).
8. M. A. Harrison, *Introduction to formal language theory* (Addison Wesley, 1978).
9. J. E. Hopcroft, J. D. Ullman, *Introduction to automata theory, languages and computation* (Addison-Wesley, 1979).
10. H. Ishizaka, *Polynomial time learnability of simple deterministic languages*, Mach. Learn. **5** (1990) 151–164.
11. M. Kearns, L. Pitt, *A polynomial-time algorithm for learning k -variable pattern languages from examples*, in: Proc. 2nd Ann. Conf. on Computational Learning Theory (ACM, 1989) 57–71.
12. M. Kearns, L. Valiant, *Cryptographic limitations on learning Boolean formulae and finite automata*, J. ACM **41** (1994) 67–95.
13. A. Marron, *Identification of pattern languages from examples and queries*, Inform. Comput. **74** (Academic Press, 1987) 91–112.
14. A. Marron, *Learning pattern languages from a single initial example and from queries*, in: Proc. 1st Ann. Workshop on Computational Learning Theory (ACM, 1988) 345–358.
15. S. Matsumoto, A. Shinohara, *Learning pattern languages using queries*, in: Proc. 3rd Euro. Conf. on Computational Learning Theory, LNAI **1208** (Springer, 1997) 185–197.
16. J. Nessel, S. Lange, *Learning erasing pattern languages with queries*, in: Proc. 11th Internat. Conf. on Algorithmic Learning Theory, LNAI **1968** (Springer, 2000) 86–100.
17. L. Pitt, M. K. Warmuth, *Prediction-preserving reduction*, J. Comput. System Sci. **41** (1990) 430–467.
18. Y. Sakakibara, *Learning context-free grammars from structural data in polynomial time*, Theor. Comput. Sci. **76** (1990) 223–242.
19. Y. Sakakibara, *On learning Smullyan's elementary formal systems: Towards an efficient learning method for context-sensitive languages*, Adv. in Soft. Sci. Tech. **2** (Academic Press, 1990) 79–101.
20. H. Sakamoto, *Language learning from membership queries and characteristic examples*, in: Proc. 6th Internat. Workshop on Algorithmic Learning Theory, LNAI **997** (Springer, 1995) 55–65.
21. H. Sakamoto, K. Hirara, H. Arimura, *Learning elementary formal systems with queries*, Technical Report DOI-TR-179, Department of Informatics, Kyushu University, 2000. Also available at <http://www.i.kyushu-u.ac.jp/doi-tr.html>.
22. E. Shamir, *On sequential languages and two classes of regular events*, Zeit. Phone., Sprach. und Kommun. **18** (1965) 61–69.
23. T. Shinohara, *Polynomial time inference of extended regular pattern languages*, in: Proc. RIMS Symposia on Software Science and Engineering, LNCS **147** (Springer, 1982) 115–127.
24. T. Shinohara, H. Arimura, *Inductive inference of unbounded unions of pattern languages from positive data*, Theor. Comput. Sci. **241** (2000) 191–209.