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

2008－AL－118 (2)
2008／5／27

極限同定性を保証する RPNIアルゴリズムの状態統合戦略について

沼井裕二 †, 小林聡 †

† 電気通信大学大学院電気通信学研究科情報工学専攻

RPNIは，Oncina と García によって提唱された，状態統合を用いて有限オートマトン (正則言語) を極限学習するアルゴリズムのひとつである．RPNIでは，状態統合の順序 (状態統合戦略) は、長さ優先の辞書式全順序を採用していた．著者らは，この戦略について考察を深めることで，RPNIさらには状態統合を利用したアルゴリズムに関する知見を得られると考えた．そこで，本論文ではまず，任意の全順序を入力して，それを状態統合戦略として利用できるようにRPNIを変更する．そして，極限学習が必ず成功する全順序の特徴づけについての考察を行う．そしてその過程で，状態統合アルゴリズムの性質についての考察も行う．

On State-Merging Strategies of RPNI Algorithm that Guarantee Identifiability in the Limit

Yuji NUMAI † Satoshi KOBAYASHI †
† Department of Computer Science,
Graduate School of Electro-Communications, The University of Electro-Communications

RPNI is an algorithm to identify regular languages in the limit by merging compatible pairs of states, which has been proposed by Oncina and García. RPNI employs the length-lexicographic total order as a state-merging strategy (SMS). The authors believe that a study on SMSs helps us comprehend RPNI and other state-merging algorithms. In this paper, RPNI is generalized so that it can employ any SMS. Moreover, the characterization of SMSs that guarantee identifiability in the limit, and some natures of state-merging algorithms are studied.

1 Introduction

RPNI is an algorithm to identify regular languages in the limit by merging compatible pairs of states, which has been proposed by Oncina and García in 1992 [1]. RPNI employs the length-lexicographic total order as a strategy. In this paper, we generalize this strategy and study the characterization of strategies that guarantee identifiability in the limit in order to comprehend some natures of RPNI and other state-merging algorithms.

RPNI is a learning algorithm in *IIL (identification in the limit) framework*, formulated by Gold [2]. First, this algorithm constructs a special finite automaton that accepts only an input positive sample of a target language. Secondly, it merges compatible pairs of states of this automaton in a specified order, which we call in this paper a *state-merging strategy (SMS)* of this algorithm. RPNI employs the length-lexicographic order as a SMS. After finishing merging states, RPNI outputs this state-merged automaton. This output is the correct minimal deterministic finite automaton (DFA) of the target language if the input sample satisfies some special condition. This condition has a monotonic property, which means that, if an input sample satisfies this condition, then any superset of this sample does so. Hence, once a given input sample of a target language satisfies this condition, RPNI does not change its conjectures, in other words, identifies the correct minimal DFA of the target language in the limit.

We have a little difficulty understanding from the verification of RPNI in [1] why RPNI with the length-lexicographic total order succeeds in identifying regular languages in the limit, for the proof is slightly short and informal. If the verification is written more precisely, then we more comprehend some natures of RPNI and other state-merging algorithms. Moreover, if we can characterize SMSs of RPNI that guarantee identifiability in the limit, then our choice of a SMS will be greatly extended, which affects the efficiency of the convergence.

In this paper, we will first generalize RPNI so that it can deal with any SMS. We will secondly study the characterization of SMSs that guarantee identifiability in the limit. In this process, we will consider some natures of RPNI and other state-merging algorithms.

There have been proposed many state-merging learning algorithms of regular languages including [3, 4], some of which use those SMSs that are not length-lexicographic. However, as far as we know, there is not established the complete characterization of SMSs of the original RPNI that guarantee identifiability in the limit.

This paper is based on the master's thesis of the first author [5], and therefore see it for more details including omitted proofs.

2 Preliminaries

We introduce basic definitions and notation needed in the sequel of this paper. We mainly employ those of Hopcroft and Ullman [6].

2.1 Finite Automata

An *alphabet* Σ is a fixed, nonempty finite set of symbols. The set of all finite strings over Σ is denoted by Σ^*. Any subset of Σ^* is called a *language* over Σ. The empty string is denoted by $\lambda \notin \Sigma$, which is contained in Σ^*. We denote the length of a string w by $|w|$. Let S be a set. The cardinality of S is denoted by $\mathrm{card}(S)$.

Let x, y be elements of Σ^*. The string x is a *prefix* of y if there exists a string $z \in \Sigma^*$ such that $xz = y$. A prefix w' of w is *proper* if $w' \neq w$. Let L be a language over Σ. We define $Pr(L) = \{ u \in \Sigma^* \mid \exists v \in \Sigma^* \text{ s.t. } uv \in L \}$. If L_1 and L_2 are languages over Σ, then it is clear that $L_1 \subseteq L_2$ implies $Pr(L_1) \subseteq Pr(L_2)$. Let u be an element in Σ^*. We define $T_L(u) = \{ v \in \Sigma^* \mid uv \in L \}$. It is also clear that, for any $u, v, w \in \Sigma^*$, $T_L(u) = T_L(v)$ implies $T_L(uw) = T_L(vw)$.

A *finite automaton* is a five-tuple $A = (Q, \Sigma, \delta, q_0, F)$ such that Q is a nonempty finite set of states, Σ is an alphabet, δ is a state transition function from $Q \times \Sigma$ to 2^Q, $q_0 (\in Q)$ is an initial state, and $F (\subseteq Q)$ is a set of final states. If $\delta(q, a)$ contains at most one element for any $q \in Q$ and any $a \in \Sigma$, then A is *deterministic*, otherwise A is *nondeterministic*. We sometimes write $\delta(p, a) = q$ for $\delta(p, a) = \{q\}$. Let $w = a_1 a_2 \cdots a_n$ $(a_i \in \Sigma)$ be an element in Σ^*, and $q, q', q_1, \ldots, q_{n-1}$ be states in Q. We write $\delta(q, w) = q'$ if $\delta(q, a_1) = q_1, \delta(q_1, a_2) = q_2, \ldots, \delta(q_{n-1}, a_n) = q'$. If $q \in \delta(p, a)$, then we say that q is an *a-successor* of p. Let q be a state in Q. The state q is called *useless* if there exist no $u, v \in \Sigma^*$ such that $\delta(q_0, u) = q$ and $\delta(q, v) \cap F = \emptyset$. A finite automaton that contains no useless states is called *reduced*. The language accepted by A, denoted by $L(A)$, is defined as $L(A) = \{ w \in \Sigma^* \mid \delta(q_0, w) \cap F = \emptyset \}$. We say that a language is *regular* if it is accepted by a finite automaton. We denote the class of regular languages by $\mathcal{REG}$.

Let $A_1 = (Q_1, \Sigma, \delta_1, q_{0_1}, F_1)$ and $A_2 = (Q_2, \Sigma, \delta_2, q_{0_2}, F_2)$. The automaton A_1 is *isomorphic* to A_2 if there exists a bijection $h\colon Q_1 \to Q_2$ satisfying the following:

$$h(q_{0_1}) = q_{0_2}; \quad h(F_1) = F_2;$$
$$\forall q \in Q_1, \forall a \in \Sigma \quad h(\delta_1(q, a)) = \delta_2(h(q), a) \ .$$

If A_1 is isomorphic to A_2, then it is clear that $L(A_1) = L(A_2)$ holds.

A *partition* π of a set Q is a set of mutually disjoint nonempty subsets of Q such that $\bigcup_{B \in \pi} B = Q$. An element of a partition is called a *block*. Let p, q be in Q. We denote the block in π containing p by $B(p, \pi)$. We write $p \cong_\pi q$ if $B(p, \pi) = B(q, \pi)$. Let $A = (Q, \Sigma, \delta, q_0, F)$, and π be a partition of Q. The *π-quotient automaton* of A is defined as $A/\pi = (Q', \Sigma, \delta', q_0', F')$, where

$$Q' \stackrel{\text{def}}{=} \pi; \quad q_0' \stackrel{\text{def}}{=} B(q_0, \pi);$$
$$F' \stackrel{\text{def}}{=} \{ B \in Q' \mid B \cap F \neq \emptyset \};$$
$$\forall B_1, B_2 \in Q' \quad \big(B_2 \in \delta'(B_1, a) \stackrel{\text{def}}{\Leftrightarrow} \exists q_1 \in B_1, \exists q_2 \in B_2 \ \ q_2 \in \delta(q_1, a)\big) \ .$$

It is clear that $L(A) \subseteq L(A/\pi)$ holds.

Let L be a nonempty regular language. The *canonical automaton* of L is defined as $A(L) = (Q, \Sigma, \delta, q_0, F)$, where

$$Q \stackrel{\text{def}}{=} \{ T_L(u) \mid u \in Pr(L) \};$$
$$q_0 \stackrel{\text{def}}{=} T_L(\lambda); \quad F \stackrel{\text{def}}{=} \{ T_L(u) \mid u \in L \};$$
$$\forall u \in Pr(L), \ \forall a \in \Sigma \text{ with } ua \in Pr(L)$$
$$\delta(T_L(u), a) \stackrel{\text{def}}{=} T_L(ua) \ .$$

It is known that $A(L)$ is the minimum-state DFA accepting L ([6, pp. 65-68]).

Let S be a finite language over Σ. The *prefix tree automaton* (*PTA*) of S is defined as $PT(S) = (Q, \Sigma, \delta, q_0, F)$, where

$$Q \stackrel{\text{def}}{=} Pr(S); \quad q_0 \stackrel{\text{def}}{=} \lambda; \quad F \stackrel{\text{def}}{=} S;$$
$$\forall u \in Pr(S), \ \forall a \in \Sigma \text{ with } ua \in Pr(S)$$
$$\delta(u, a) \stackrel{\text{def}}{=} ua \ .$$

It is clear that $PT(S)$ exactly accepts S.

Let g be a grammar. By $L(g)$ we denote the language generated by g.

2.2 Total Order

Let S be a set, a and b be elements of S, and $\preceq$ be a relation on S. The relation $\preceq$ is called an *order* if it is reflexive, antisymmetric and transitive. We write $a \prec b$ if $a \preceq b$ and $a \neq b$ hold. Let $\preceq$ be an order on a set S. The order $\preceq$ is *total* if $a \preceq b$ or $b \preceq a$ holds for any distinct a and b in S.

Let X be a subset of a set S, m be an element of X, and $\preceq$ be a total order on S. The element m is *the greatest one* of X for $\preceq$, denoted by $\max_{\preceq}(X)$ (respectively, *the least one* of X for $\preceq$, denoted by $\min_{\preceq}(X)$), if $x \preceq m$ (respectively, $m \preceq x$) holds for any $x \in X$. Let $\preceq$ be a total order on Σ^*, and x be an element in Σ^*. We define $ip_{\preceq}(x) = x_p$, where x_p in Σ^*, $x_p \prec x$, and there is no y in Σ^* such that $x_p \prec y \prec x$.

Let $\preceq$ be a total order on Σ^*, and let B_1 and B_2 be subsets of Σ^*. We write $B_1 \prec B_2$ if $u \prec v$ holds for any $u \in B_1$ and any $v \in B_2$. Let π be a partition of a set, and B_1 and B_2 be blocks in π. The new partition obtained by merging B_1 with B_2, denoted by $J(\pi, B_1, B_2)$, is defined as $J(\pi, B_1, B_2) = (\pi - \{B_1, B_2\}) \cup \{B_1 \cup B_2\}$.

2.3 IIL Framework

Let L be a language over Σ. A *sample* of L is denoted by a pair $S = (S_+, S_-)$ that satisfies $S_+ \subseteq L$ and $S_- \subseteq \Sigma^* - L$. We call S_+ and S_-, respectively, a *positive sample* and a *negative sample* of L. Let S_+, S_-, S'_+, and S'_- be sets. We say that $S' = (S'_+, S'_-)$ is a *superpair* of $S = (S_+, S_-)$, which we denote by $S \sqsubseteq S'$, if $S_+ \subseteq S'_+$ and $S_- \subseteq S'_-$ hold. Let L be a language over Σ, and $\sigma = (w_1, l_1), (w_2, l_2), \ldots$ be an infinite sequence such that $w_i \in \Sigma^*$ and $l_i \in \{0, 1\}$ $(i = 1, 2, \ldots)$. An infinite sequence σ is a *complete presentation* of L if the following conditions hold:

1. $\{w_1, w_2, \ldots\} = \Sigma^*$,
2. $\forall i \geq 1 \quad (l_i = 1) \Leftrightarrow (w_i \in L)$.

A *learning machine* M is an algorithmic device which receives a complete presentation of a target language L and outputs an infinite sequence $g_1, g_2, \ldots$ of grammars. We say that M *identifies* L *in the limit* if, for any complete presentation σ of L, M with an input σ outputs an infinite sequence $g_1, g_2, \ldots$ of grammars such that

$$\exists n_0 \in \mathbf{N} \text{ s.t. } \forall n \geq n_0 \quad L(g_n) = L \ .$$

Let $\mathcal{C}$ be a class of languages. We also say that M identifies $\mathcal{C}$ in the limit if M identifies any L in $\mathcal{C}$ in the limit.

3 k-state Infinite Languages

In this section, we introduce a new interesting language class related to finite automata and their states. It plays an important role in characterizing SMSs that guarantee identifiability in the limit.

Let $A = (Q, \Sigma, \delta, q_0, F)$ be a reduced DFA, and L be a regular language over Σ. We define

$$Q^{\infty}(L, A) = \{\, q \in Q \mid \mathrm{card}(\{\, w \in L \mid \delta(q_0, w) = q \,\}) = \infty \,\}.$$

We say that L is *k-state infinite* for A if $\mathrm{card}(Q^{\infty}(L, A)) = k$, and that L is k-state infinite if L is k-state infinite for any reduced DFA. Moreover, we say that L is up-to-k-state infinite if

$$\neg\exists A \text{ s.t. } \exists k' \text{ with } k' > k \text{ s.t. } L \text{ is } k'\text{-state-infinite for } A \ .$$

Let $\preceq$ be a total order on Σ^*, and u be an element in Σ^*. We define $S(\preceq, u) = \{\, v \in \Sigma^* \mid v \preceq u \,\}$. The following lemma and its corollary hold.

Lemma 1. *Let $\preceq$ be a total order on Σ^*, u and v be in Σ^*, and $A = (Q, \Sigma, \delta, q_0, F)$ be a reduced DFA. We have*

$$Q^{\infty}(S(\preceq, u), A) = Q^{\infty}(S(\preceq, v), A) \ .$$

□

Corollary 1. *Let $\preceq$ be a total order on Σ^*, u and v be elements in Σ^*, and $A = (Q, \Sigma, \delta, q_0, F)$ be a reduced DFA. We have*

$$S(\preceq, u) \text{ is up-to-}k\text{-state infinite for } A \Leftrightarrow S(\preceq, v) \text{ is up-to-}k\text{-state infinite for } A \ .$$

□

4 Revision of RPNI

We propose Algorithm 1 by generalizing the original RPNI. Oncina and García have inputted the length-lexicographic total order on Σ^* into RPNI for choosing a state w_i to be merged on the i-th "of the for-loop" [1], while we can input any total order (SMS) on Σ^* into our algorithm. Moreover, the original RPNI choose a state to be merged "in if-statements" in the length-lexicographic order, while our algorithm can choose any state which can be merged.

5 Correctness of RPNI and Road to Characterization of SMSs

We will now introduce a special class of SMSs. By $\mathcal{O}_{\mathrm{R}}$ we denote the class of total orders $\preceq$ on Σ^* such that

Algorithm 1 RPNI algorithm (revised)

Input: A sample $S = (S_+, S_-)$ of a target regular language L over Σ, and a total order $\preceq$ on Σ^*

Output: A_0/π_r

Let $Pr(S_+) = \{w_0, w_1, \ldots, w_r\}$ and suppose that $w_0 \prec w_1 \prec \cdots \prec w_r$ holds;
$A_0 := PT(S_+)$;
  // constructs the PTA of S_+
$\pi_0 := \{\{w_0\}, \{w_1\}, \ldots, \{w_r\}\}$;
for $i = 1$ to r **do** // chooses a state $\{w_i\}$
  if $\exists B_\alpha, B_\beta \in \pi_{i-1}$, $\exists a \in \Sigma$ s.t. $(B_\beta \prec \{w_i\})$ $\wedge$ (B_β and $\{w_i\}$ are a-successors of B_α) $\wedge$ $\big(S_- \cap L(A_0/J(\pi_{i-1}, B_\beta, \{w_i\})) = \emptyset\big)$ **then**
    $\pi_i := J(\pi_{i-1}, B_\beta, \{w_i\})$;
      // merges B_β with $\{w_i\}$
  else if $\exists B_\gamma \in \pi_{i-1}$ s.t. $(B_\gamma \prec \{w_i\})$ $\wedge$ $\big(S_- \cap L(A_0/J(\pi_{i-1}, B_\gamma, \{w_i\})) = \emptyset\big)$ **then**
    $\pi_i := J(\pi_{i-1}, B_\gamma, \{w_i\})$;
      // merges B_γ with $\{w_i\}$
  else
    $\pi_i := \pi_{i-1}$;
      // doesn't merge a pair of states
  end if
end for
return A_0/π_r;
  // outputs the state-merged automaton

1. $S(\preceq, \lambda)$ is up-to-1-state infinite,
2. $\exists w_b \in \Sigma^*$ s.t. $\forall w \in \Sigma^*$ with $w_b \preceq w$ $\big(\neg\exists w_p \in \Sigma^*$ s.t. $(w \prec w_p) \wedge (\exists a \in \Sigma$ s.t. $w_p a = w)\big)$.

In the definition above, an element w_b satisfying the condition 2 is called a *boundary* of $\preceq$, denoted by $bd_\preceq$.

5.1 Definitions and Notation

Let L be a regular language over Σ, $A(L) = (Q, \Sigma, \delta, q_0, F)$ be the canonical automaton of L, and $\preceq$ be a total order on Σ^*. Let $pp = ip_\preceq(\min_\preceq(PSP_\preceq(L)))$. We define some sets as follows:

$$PSP_\preceq(L) = \{\, w \in Pr(L) \mid \neg\exists v \in \Sigma^* \text{ s.t. } T_L(v) = T_L(w) \ \wedge\ v \prec w \,\};$$

$$SP_\preceq(L) = \begin{cases} PSP_\preceq(L) & \text{if (a) holds} \\ PSP_\preceq(L) \cup \{pp\} & \text{if (b) holds} \\ \text{undefined} & \text{otherwise,} \end{cases}$$

where

(a) $S(\preceq, \lambda)$ is 0-state infinite for $A(L)$,

(b) $S(\preceq, \lambda)$ is 1-state infinite for $A(L)$;

$$N_\preceq(L) = \{\, wa \in Pr(L) \mid w \in SP_\preceq(L) \wedge a \in \Sigma \,\} \cup \{\lambda\};$$

$$SN_\preceq(L) = SP_\preceq(L) \cup N_\preceq(L);$$

$$ESN_\preceq(L) = \{\, w \in \Sigma^* \mid \min_\preceq(SN_\preceq(L)) \preceq w \preceq \max_\preceq(\{\max_\preceq(SN_\preceq(L)), bd_\preceq\}) \,\} .$$

The following holds:

$$\preceq \in \mathcal{O}_{\mathrm{R}} \ \Rightarrow\ SP_\preceq(L) \text{ is defined and } PSP_\preceq(L) \subseteq SP_\preceq(L);$$
$$SN_\preceq(L) \subseteq ESN_\preceq(L) \ .$$

Let $S = (S_+, S_-)$ be a sample of a regular language L over Σ and let $\preceq$ be a total order on Σ^*. The sample S is *complete* for $\preceq$ if the following conditions hold:

1. $\forall u \in SN_\preceq(L)$ $\big(u \in Pr(S_+)\big) \ \wedge\ (u \in L \ \Rightarrow\ u \in S_+)$,
2. $\forall u \in SP_\preceq(L)$, $\forall v \in ESN_\preceq(L)$ $\big(T_L(u) \neq T_L(v) \ \Rightarrow$ either $(\exists u' \in \Sigma^*$ s.t. $uu' \in S_+ \wedge vu' \in S_-)$ or $(\exists v' \in \Sigma^*$ s.t. $vv' \in S_+ \wedge uv' \in S_-)$ holds$\big)$.

Let S_1 and S_2 be samples of a regular language L such that $S_1 \sqsubseteq S_2$ holds. It is clear that, if S_1 is complete for $\preceq$, then S_2 is also complete for $\preceq$. In other words, the completeness for $\preceq$ has a monotonic property.

5.2 Proof of Sufficiency

In the end of this subsection we will prove the following theorem:

Theorem 1. *For any $\preceq$ in $\mathcal{O}_{\mathrm{R}}$, Algorithm 1 identifies $\mathcal{REG}$ in the limit (later proved).*

□

Let $\preceq$ be a total order in $\mathcal{O}_{\mathrm{R}}$, $S = (S_+, S_-)$ be a sample of a regular language L over Σ, $A_0 = PT(S_+)$, $A(L) = (Q, \Sigma, \delta, p_0, F)$ be the canonical automaton of L, and let $Pr(S_+) = \{w_0, w_1, \ldots, w_r\}$ where we assume that $w_0 \prec w_1 \prec \cdots \prec w_r$ holds. Let $\pi_0, \pi_1, \ldots, \pi_r$ be the partition sequence produced by Algorithm 1 with the inputs S and $\preceq$. These are assumed in this entire subsection.

Lemma 2. *Suppose that S is complete for $\preceq$, that π be a partition of $Pr(S_+)$, and that $S_- \cap L(A_0/\pi) = \emptyset$. Consider $u \in ESN_\preceq(L)$ and $v \in SP_\preceq(L)$ such that $u \cong_\pi v$. We have*

$$T_L(u) = T_L(v) \ .$$

□

Lemma 3. *Let π be a partition of $Pr(S_+)$ such that, for any u and v in $Pr(S_+)$, $u \cong_\pi v$ implies $T_L(u) = T_L(v)$. Consider any w and w' in $Pr(S_+)$ such that $T_L(w) = T_L(w')$. We have*

$$S_{-} \cap L(A_0/J(\pi, B(w,\pi), B(w',\pi))) = \emptyset \ .$$

□

Lemma 4. *Suppose that S is complete for $\preceq$. It holds that $w_s = \min_{\preceq}(SP_{\preceq}(L))$ $(0 \leq s \leq r)$. For any i $(0 \leq i \leq r)$, we have*

$$\forall a, b \leq i \quad \left(w_a \cong_{\pi_i} w_b \ \Leftrightarrow \ T_L(w_a) = T_L(w_b)\right), \quad (1)$$

$$i \geq s \ \Rightarrow \ \forall j \ (j \leq i) \ \ B(w_j, \pi_i) \text{ contains exactly one element of } SP_{\preceq}(L) \ . \quad (2)$$

Proof. We will prove the claims by mathematical induction on i.

(Base Step) Consider the case $i = 0$.

The equation (1) clearly holds.

Suppose that $i \geq s$. Hence, $s = 0$. Note that $w_0 = w_s = \min_{\preceq}(SP_{\preceq}(L)) \in SP_{\preceq}(L)$. Hence the block $B(w_0, \pi_0) = \{w_0\}$ contains exactly one element w_0 in $SP_{\preceq}(L)$.

(Induction Step) Suppose that (1) and (2) hold for $i = k - 1$. Note that $k = i + 1 \geq 1$.

(I) Suppose that the condition of the first if-statement holds on the k-th run of the for-loop. There exist $B_\alpha \in \pi_{k-1}$, $B_\beta \in \pi_{k-1}$ and $a \in \Sigma$ such that

$$B_\beta \text{ and } \{w_k\} \text{ are } a\text{−successors of } B_\alpha, \quad (A)$$
$$B_\beta \prec \{w_k\}, \quad (B)$$
$$S_{-} \cap L(A_0/J(\pi_{k-1}, B_\beta, \{w_k\})) = \emptyset \ . \quad (C)$$

(Claim I-i) The equation (1) holds for $i = k$.

Proof of Claim I-i. It suffices to prove that, for any $w_j \in Pr(S_+)$ $(0 \leq j \leq k)$, $w_j \in B(w_k, \pi_k)$ if and only if $T_L(w_j) = T_L(w_k)$. The condition (A) implies that there exist $w, w' \in B_\alpha$ such that $wa = w_k$ and $w'a \in B_\beta$. The inductive hypothesis (IH) (1) implies that $T_L(w') = T_L(w)$. Hence, $T_L(w'a) = T_L(wa) = T_L(w_k)$. Considering $w_j \in Pr(S_+)$ for $0 \leq j \leq k$. If $w_j \in \{w_k\}$, then it is clear that $T_L(w_j) = T_L(w_k)$. If $w_j \in B_\beta$, then the IH (1) implies that $T_L(w_j) = T_L(w'a) = T_L(w_k)$. Hence, if $w_j \in B(w_k, \pi_k) = B_\beta \cup \{w_k\}$, then $T_L(w_j) = T_L(w_k)$. If $w_j \notin B(w_k, \pi_k) = B_\beta \cup \{w_k\}$, then it holds that $w_j \notin B_\beta$, which means that $T_L(w_j) \neq T_L(w'a) = T_L(w_k)$. Now it follows that, for any $w_j \in Pr(S_+)$ $(0 \leq j \leq k)$, $w_j \in B(w_k, \pi_k)$ if and only if $T_L(w_j) = T_L(w_k)$.

[*End of Claim I-i*]

(Claim I-ii) The equation (2) holds for $i = k$.

Proof of Claim I-ii. First, suppose that $k = s$. Note that $k = s \geq 1$ and $w_s = \min_{\preceq}(SP_{\preceq}(L))$. Hence, $S(\preceq, \lambda)$ is 1-state infinite for $A(L)$. We deduce from the definition of 1-state-infiniteness that $T_L(w_0) = \cdots = T_L(w_k) = T_L(w_s)$ holds. We deduce from the IH (1) that $w_0 \cong_{\pi_{k-1}} \cdots \cong_{\pi_{k-1}} w_{k-1}$. The condition (B) implies that $B_\beta \in \{B(w_0, \pi_{k-1}), \ldots, B(w_{k-1}, \pi_{k-1})\}$. Hence, after $\{w_k\}$ is merged with B_β, it holds that $w_0 \cong_{\pi_k} \cdots \cong_{\pi_k} w_k$. Hence, the block $B(w_0, \pi_k) = \cdots = B(w_k, \pi_k)$ contains exactly one element of $SP_{\preceq}(L)$, $w_k = w_s = \min_{\preceq}(SP_{\preceq}(L))$.

Secondly, suppose that $k > s$. It holds that $k-1 \geq s$, which together with the IH (2) implies that, for any j $(j \leq k-1)$, $B(w_j, \pi_{k-1})$ contains exactly one element of $SP_{\preceq}(L)$. Hence, with the condition (B), It holds that there exists $w_{sp} \in B_\beta$ such that $w_{sp} \in SP_{\preceq}(L)$. Suppose that $w_k \in SP_{\preceq}(L)$. We deduce from the definition of $SP_{\preceq}(L)$ that $T_L(w_{sp}) \neq T_L(w_k)$. Note that S is complete for $\preceq$, $w_{sp} \in SP_{\preceq}(L)$, $w_k \in SP_{\preceq}(L) \subseteq ESN_{\preceq}(L)$ and (C) hold. By using Lemma 2, we obtain $T_L(w_{sp}) = T_L(w_k)$, which is a contradiction. Hence $w_k \notin SP_{\preceq}(L)$, which implies that the block $B(w_k, \pi_k) = B_\beta \cup \{w_k\}$ contains exactly one element of $SP_{\preceq}(L)$, w_{sp}.

[*End of Claim I-ii*]

(II) Suppose that the condition of the first if-statement does not hold and that of the second holds on the k-th run of the for-loop. There exists $B_\gamma \in \pi_{k-1}$ such that

$$B_\gamma \prec \{w_k\}, \quad (D)$$
$$S_{-} \cap L(A_0/J(\pi_{k-1}, B_\gamma, \{w_k\})) = \emptyset \ . \quad (E)$$

(Claim II-i) The equation (1) holds for $i = k$.

Proof of Claim II-i. It suffices to prove that, for any $w_j \in Pr(S_+)$ $(0 \leq j \leq k)$, $w_j \in B(w_k, \pi_k)$ if and only if $T_L(w_j) = T_L(w_k)$.

First, suppose that $k \leq s$. Note that $k = s \geq 1$ and $w_s = \min_{\preceq}(SP_{\preceq}(L))$. Hence, $S(\preceq, \lambda)$ is 1-state infinite for $A(L)$. We deduce from the definition of 1-state-infiniteness that $T_L(w_0) = \cdots = T_L(w_k) = \cdots = T_L(w_s)$ holds. This and the IH (1) implies that $w_0 \cong_{\pi_{k-1}} \cdots \cong_{\pi_{k-1}} w_{k-1}$. The condition (D) implies that $B_\gamma \in \{B(w_0, \pi_{k-1}), \ldots, B(w_{k-1}, \pi_{k-1})\}$. Hence, after $\{w_k\}$ is merged with B_γ, it holds that $w_0 \cong_{\pi_k} \cdots \cong_{\pi_k} w_k$, which implies that (1) holds for $i = k$.

Secondly, suppose that $k > s$.

To prove that $w_k \in ESN_{\preceq}(L)$, let us suppose the opposite and see what happens. We deduce from $N_{\preceq}(L) \subseteq ESN_{\preceq}(L)$ that $w_k \notin N_{\preceq}(L)$. Hence $w_k \neq \lambda$, which means that there

exist $w' \in \Sigma^*$, $B' \in \pi_{k-1}$, and $a \in \Sigma$ such that $w' \in B'$ and $w'a = w_k$. Suppose that $w' \in SP_\preceq(L)$. It holds that $w'a = w_k \in N_\preceq(L)$, which is a contradiction. Hence $w' \notin SP_\preceq(L)$. The condition $k > s$ implies that $k-1 \geq s$, which together with the IH (2) that, for any j $(j \leq k-1)$, $B(w_j, \pi_{k-1})$ contains exactly one element of $SP_\preceq(L)$. We want to know whether $w_k \prec \min_\preceq(ESN_\preceq(L))$ or $\max_\preceq(ESN_\preceq(L)) \prec w_k$. If $w_k \prec \min_\preceq(ESN_\preceq(L))$, then $w_s = \min_\preceq(SP_\preceq(L)) \prec w_k \prec \min_\preceq(ESN_\preceq(L)) = \min_\preceq(SN_\preceq(L))$, which contradicts $SP_\preceq(L) \subseteq SN_\preceq(L)$. Hence, it holds that $\max_\preceq(ESN_\preceq(L)) = \max_\preceq(\{\max_\preceq(SN_\preceq(L)), bd_\preceq\}) \prec w_k$, which means that $bd_\preceq \prec w_k$. We want to show $w' \prec w_k = w'a$. Note that $bd_\preceq \prec w_k = w'a$. We deduce from the definition of $\mathcal{O}_\mathrm{R}$ that $w' \prec w'a = w_k$. This and the IH (2) implies that there exists $w_{sp_0} \in B(w', \pi_{k-1})$ such that $w_{sp_0} \in SP_\preceq(L)$. The IH (1) implies that $T_L(w_{sp_0}) = T_L(w')$. Hence, $T_L(w_{sp_0}a) = T_L(w'a) = T_L(w_k)$. Because $w_k \in Pr(S_+) \subseteq Pr(L)$, it holds that $T_L(w_{sp_0}a) = T_L(w_k) \neq \emptyset$. Hence $w_{sp_0}a \in Pr(L)$, which together with $w_{sp_0} \in SP_\preceq(L)$ implies $w_{sp_0}a \in N_\preceq(L) \subseteq SN_\preceq(L)$. By using the completeness for $\preceq$ of S, we obtain $w_{sp_0}a \in Pr(S_+)$. Because $w_{sp_0}a \in SN_\preceq(L) \subseteq ESN_\preceq(L)$ and $\max_\preceq(ESN_\preceq(L)) \prec w_k$, it holds that $w_{sp_0}a \prec w_k$. Note that $T_L(w_{sp_0}a) = T_L(w_k)$ and the IH (1) holds. By using Lemma 3, we have $S_- \cap L(A_0/J(\pi_{k-1}, B(w_{sp_0}a, \pi_{k-1}), \{w_k\})) = \emptyset$. This means that the condition of the first if-statement is satisfied, which is a contradiction. Hence $w_k \in ESN_\preceq(L)$.

The condition (D) implies there exists $w_{sp_1} \in B_\gamma$ such that $w_{sp_1} \in SP_\preceq(L)$. Note that S is complete for $\preceq$, $w_{sp_1} \in SP_\preceq(L)$, $w_k \in ESN_\preceq(L)$, $w_{sp_1} \cong_{\pi_k} w_k$, and (E) holds. By using Lemma 2, we obtain $T_L(w_{sp_1}) = T_L(w_k)$. In a similar way as in the proof of Claim I-i, we deduce that, for any $w_j \in Pr(S_+)$ $(0 \leq j \leq k)$, $w_j \in B(w_k, \pi_k)$ if and only if $T_L(w_j) = T_L(w_k)$.

[*End of Claim II-i*]

(Claim II-ii) The equation (2) holds for $i = k$.

Proof of Claim II-ii. We can prove this claim in a similar way as in the proof of Claim I-ii.

[*End of Claim II-ii*]

(III) Suppose that the conditions of the first and second if-statements do not hold on the k-th run of the for-loop.

(Claim III-i) The equation (1) holds for $i = k$.

Proof of Claim III-i. Because $\pi_k = \pi_{k-1}$ holds, the claim also holds.

[*End of Claim III-i*]

(Claim III-ii) The equation (2) holds for $i = k$.

Proof of Claim III-ii. First, suppose that $k = s$. Because $k \geq 1$, there exist w_{k-1}. It holds that $S(\preceq, \lambda)$ is 1-state infinite for $A(L)$. We deduce from the definition of 1-state-infiniteness that, for any w with $w \preceq w_s = \min_\preceq(SP_\preceq(L))$, $T_L(w) = T_L(w_s)$ holds. Hence, it holds that $T_L(w_s) = T_L(w_k) = T_L(w_{k-1})$. However, we deduce from $w_{k-1} \not\cong_{\pi_k} w_k$ that $T_L(w_{k-1}) \neq T_L(w_k)$, which is a contradiction.

Secondly, suppose that $k > s$. Let B' be a block in π_{k-1} such that $B' \prec \{w_k\}$. It holds that $S_- \cap L(A_0/J(\pi_{k-1}, B', \{w_k\})) \neq \emptyset$. Note that the IH (1) holds. We deduce from Lemma 3 that there exists no $w' \in B'$ such that $T_L(w') = T_L(w_k)$. This means that there exists no $w'' \in Pr(S_+) \subseteq Pr(L)$ such that $T_L(w'') = T_L(w_k)$ and $w'' \prec w_k$. We deduce from the definition of $PSP_\preceq(L)$ that $w_k \in PSP_\preceq(L) \subseteq SP_\preceq(L)$. Hence, the block $B(w_k, \pi_k) = \{w_k\}$ contains exactly one element w_k in $SP_\preceq(L)$.

[*End of Claim III-ii*]

The discussions (I), (II) and (III) complete the induction step.

□

Let S be complete for $\preceq$, and let $A_0/\pi_r = (Q_r, \Sigma, \delta_r, q_{0,r}, F_r)$ be the output of Algorithm 1 with the input S and $\preceq$. We define the mapping $h\colon Q_r \to Q$ as $h(B(w, \pi_r)) = T_L(w)$. We deduce from Lemma 4(1) that h is well-defined.

Lemma 5. *h is a bijection.*

Proof. We will first show that h is an injection. Suppose that $h(B(w, \pi_r)) = h(B(w', \pi_r))$ for any w, $w' \in Pr(S_+)$. We deduce from the definition of h that $T_L(w) = T_L(w')$. Suppose that $w \not\cong_{\pi_r} w'$. Lemma 4(1) implies that $T_L(w) \neq T_L(w')$, which is a contradiction. Hence it holds that $w \cong_{\pi_r} w'$, which means that h is an injection.

We will secondly show that h is a surjection. We deduce from the definition of $SP_\preceq(L)$ that, for any $w \in Pr(L)$, there exists $w_{sp} \in SP_\preceq(L)$ such that $T_L(w) = T_L(w_{sp})$. Because S is complete for $\preceq$, $SP_\preceq(L) \subseteq SN_\preceq(L) \subseteq Pr(S_+)$ holds. Hence, for any $w_{sp} \in SP_\preceq(L) \subseteq Pr(S_+)$, there exists $B(w_{sp}, \pi_r) \in Q_r$. We have, for any $w \in Pr(L)$, $T_L(w) = T_L(w_{sp}) = h(B(w_{sp}, \pi_r))$ holds, which means that h is a surjection.

□

Lemma 6. *$L(A_0/\pi_r) = L$.*

Proof. We now want to show that A_0/π_r is isomorphic to $A(L)$. The following three claims are needed (proofs omitted):

1. $h(q_{0_r}) = p_0$.
2. $h(F_r) = F$.
3. Let $q \in Pr(S_+)$, $B(q, \pi_r) \in Q_r$, and $a \in \Sigma$. It holds that $h(\delta_r(B(q, \pi_r), a)) = \delta(h(B(q, \pi_r)), a)$.

We deduce from Lemma 5 and the above claims that A_0/π_r is isomorphic to $A(L)$.

□

We at last establish the following theorem.

Theorem 1. *For any $\preceq$ in $\mathcal{O}_{\mathrm{R}}$, Algorithm 1 identifies $\mathcal{REG}$ in the limit.*

Proof. Let S be complete for $\preceq$. Lemma 6 implies that Algorithm 1 with the inputs $\preceq$ and S outputs the canonical automaton of L. Remark that the completeness for $\preceq$ has a monotonic property. Hence, Algorithm 1 with the inputs $\preceq$ and any sample S' such that $S \sqsubseteq S'$ outputs the canonical automaton of L.

□

5.3 Road to Proof of Necessity

We want to show that, for any $\preceq$ not in $\mathcal{O}_{\mathrm{R}}$, Algorithm 1 can not always identify $\mathcal{REG}$ in the limit. The claim above is half proved (Lemma 10) but is not completely proved yet.

5.3.1 Proof of Lemma 10

Let $\mathcal{O}_{\mathrm{S}}$ be the set of total orders on Σ^* such that the first condition of $\mathcal{O}_{\mathrm{R}}$ does not hold, and let $\preceq$ be a total order in $\mathcal{O}_{\mathrm{S}}$.

Lemma 7. *There exists $L \in \mathcal{REG}$ such that there exists an infinite pair sequence $(x_1, y_1), (x_2, y_2), \ldots$ that satisfies*

$$
\begin{aligned}
&x_1, y_1 \in S(\preceq, \min_{\preceq}(PSP_{\preceq}(L)));\\
&T_L(x_1) \neq T_L(y_1);\\
&T_L(x_1) = T_L(x_2) = \cdots;\\
&T_L(y_1) = T_L(y_2) = \cdots;\\
&\cdots \prec x_3 \prec \cdots \prec y_3 \prec \cdots \prec x_2 \prec \cdots\\
&\quad \prec y_2 \prec \cdots \prec x_1 \prec \cdots \prec y_1 \prec \cdots\\
&\quad \prec \min_{\preceq}(PSP_{\preceq}(L)) \prec \cdots;\\
&|y_1| < |x_1| < |y_2| < |x_2| < \cdots \ .
\end{aligned}
$$

□

We define

$$
\begin{aligned}
m_i = \min\{m_{i-1}\} \cup \{\, |w| \mid w \in \Sigma^* \ \wedge\\
w \prec y_i \ \wedge \ |w| < |y_i| \,\} \quad \text{for } i \geq 1,
\end{aligned}
$$

where $m_0 = 0$.

Lemma 8. $m_1 \leq m_2 \leq m_3 \leq \cdots$.

□

Lemma 9. *An infinite sequence σ_{S} enumerated as follows is a complete presentation of L in Lemma 7.*

for *$i = 1$ to ∞* ***do***
enumerate all the elements $w \in \Sigma^*$ such that $|w| < m_i$ and w is not enumerated yet.

enumerate two elements $x'_i, y'_i \in \Sigma^*$ such that $x'_i, y'_i \in L$ and x_i, y_i are a proper prefix of x'_i, y'_i, respectively.
end for

□

Lemma 10. *For any $\preceq$ in $\mathcal{O}_{\mathrm{S}}$, Algorithm 1 can not always identify $\mathcal{REG}$ in the limit.*

Proof. Assume that we enumerate elements of Σ^* by using σ_{S} and input them into Algorithm 1.

Suppose that we are on the k-th run of the for-loop in σ_{S}. The positive and negative samples are denoted as follows:

$$
\begin{aligned}
S_+ &= \{x'_1, \ \ldots \ , x'_k, y'_1, \ \ldots \ , y'_k\} \cup\\
&\quad \{\, w \in \Sigma^* \mid |w| < m_k \ \wedge \ w \in L \,\},\\
S_- &= \{\, w \in \Sigma^* \mid |w| < m_k \ \wedge \ w \notin L \,\} \ .
\end{aligned}
$$

It holds that $x'_k, y'_k \in S_+$ and hence their prefixes x_k, y_k are in $Pr(S_+)$, that is, x_k, y_k are states of $PT(S_+)$. For any element w such that $w \prec y_k$, it holds that $|w| \geq m_k$. This means that merging w with w' such that $w \prec y_k$ and $w' \prec y_k$ is successful. Hence, Algorithm 1 succeeds in merging x_k with y_k. Because $T_L(x_k) \neq T_L(y_k)$, the output automaton is different from $A(L)$.

It follows that, Algorithm 1 infinitely merges a pair of states that must not be merged.

□

We will be able to show the necessity if we can prove that, for any total order such that the second condition of $\mathcal{O}_{\mathrm{R}}$ does not hold, Algorithm 1 can not always identify $\mathcal{REG}$ in the limit.

6 Conclusions

We deduce from the discussions in Section 5 that we can characterize SMSs that guarantee identifiability in the limit if the following claim is established:

for any total order such that the second condition of $\mathcal{O}_{\mathrm{R}}$ does not hold, Algorithm 1 can not always identify $\mathcal{REG}$ in the limit.

6.1 Natures of State-Merging Algorithms

We consider why RPNI with a SMS in $\mathcal{O}_{\mathrm{R}}$ can identify $\mathcal{REG}$ in the limit. Let $\preceq$ be a total order in $\mathcal{O}_{\mathrm{R}}$, L be a regular language over Σ, $S = (S_+, S_-)$ be a sample of L, and $A(L) = (Q, \Sigma, \delta, q_0, F)$ be the canonical automaton of L.

We will first consider some natures of the first condition of $\mathcal{O}_{\mathrm{R}}$. Because $\preceq \in \mathcal{O}_{\mathrm{R}}$ holds, $SP_{\preceq}(L)$ is defined. We deduce from the definition of $SP_{\preceq}(L)$ and Q that there is a one-to-one correspondence between $SP_{\preceq}(L)$ and Q. The proof of Lemma 4 implies that an element in $SP_{\preceq}(L)$ serves as a marker representing a state in Q when RPNI merges compatible pairs of states. It follows that $PSP_{\preceq}(L)$ and $SP_{\preceq}(L)$ may play an important role in RPNI and other state-merging algorithms.

We will next consider those of the second. Let $w, w' \in \Sigma^*$ and $a \in \Sigma$ such that $w'a = w$. If $bd_{\preceq} \prec w$ holds, then $w' \prec w$ holds. We deduce from the proof of Lemma 4 that w' has nondeterministic transitions of a, which means that these transitions are removed in the first if-then statement of the for-loop in RPNI. In conclusion, by using the second condition of $\mathcal{O}_{\mathrm{R}}$, RPNI can fold up $PT(S_+)$ from inside into $A(L)$.

6.2 Future Works

Some of our future works are the following.

We must extend our results to characterize SMSs that guarantee identifiability in the limit. We believe that Algorithm 1 identifies $\mathcal{REG}$ in the limit if and only if an input total order of Algorithm 1 is in $\mathcal{O}_{\mathrm{R}}$.

One of the other works is to characterize SMSs of RPNI variants. There exist some variants of RPNI, one of which [3, 7, 8] gives priority to remove nondeterministic transitions generated when merging compatible pairs of states. Such a change of an algorithm also changes the characterization of SMSs of the algorithm. We believe that the characterization in that algorithm [3, 7, 8] is more sophisticated than that of our algorithm.

References

[1] J. Oncina and P. García, "Inferring regular languages in polynomial updated time," *Pattern Recognition and Image Analysis, Series in Machine Perception & Artificial Intelligence*, vol. 1, pp. 49–61, 1992.

[2] E. M. Gold, "Language identification in the limit," *Information and Control*, vol. 10, pp. 447–474, 1967.

[3] K. J. Lang, B. A. Pearlmutter, and R. Price, "Results of the abbadingo one DFA learning competition and a new evidence driven state merging algorithm," *Proceedings of the Fourth International Colloquium on Grammatical Inference (ICGI-98)*, pp. 1–12, 1998.

[4] C. D. L. Higuera, J. Oncina, and E. Vidal, "Identification of DFA: data-dependent versus data-independent algorithms," *Proceedings of the Third International Colloquium on Grammatical Inference (ICGI-96)*, pp. 311–325, 1996.

[5] Y. Numai, "A study on state-merging strategies of RPNI algorithm," Master's thesis, Department of Computer Science, Graduate School of Electro-Communications, The University of Electro-Communications, 2008.

[6] J. Hopcroft and J. Ullman, *Introduction to Automata Theory, Languages and Computation.* Addison-Wesley, 1979.

[7] P. Dupont, "Incremental regular inference," *Proceedings of the Third International Colloquium on Grammatical Inference (ICGI-96)*, pp. 222–237, 1996.

[8] R. Parekh and V. Honavar, "Learning DFA from simple examples," *Proceedings of the Eighth International Workshop on Algorithmic Learning Theory (ALT-97), Lecture Notes in Computer Science*, vol. 1316, pp. 116–131, 1997.