Master Thesis

Minimization of Nondeterministic Finite Automata

Supervisor: Professor Shuzo YAJIMA

Department of Information Science
Faculty of Engineering
Kyoto University

Hiroaki SENGOKU

February 14, 1992

Minimization of Nondeterministic Finite Automata

Hiroaki SENGOKU

Abstract

A nondeterministic finite automaton is the essential concept of the theory of formal language. It is also closely related to the coding theory, and plays an important role in various applications. It is, therefore, very important to investigate and understand the properties of nondeterministic finite automata. In this thesis, the minimum nondeterministic finite automata equivalent to the given automaton are discussed, and the minimization algorithm is presented.

In the case of deterministic finite automata, by merging equivalent states of the given automaton, the unique reduced deterministic finite automaton can be obtained. Furthermore, the reduced deterministic finite automaton is the minimum state deterministic automaton. However in the case of nondeterministic finite automata, the reduced automaton is not always minimum. It is obvious that there are only finite number of nondeterministic finite automata which have fewer states than the given automaton. Therefore we can always find the minimum equivalent automata by exhaustive search. However this method is not practical.

The first non-exhaustive search algorithm for minimizing nondeterministic finite automata was proposed by Kameda and Weiner[3]. In their algorithm, they make a canonical form of the given automaton, called a reduced automaton matrix, and synthesize the

minimum nondeterministic automaton from it. However, all the synthesized automata are not equivalent to the given one. Thus, if the synthesized automaton is not equivalent, they must search another automaton from the matrix.

In this thesis, we present more effective algorithm and prove its correctness. That is, all the constructed automata are equivalent to the given one, and all the minimum equivalent automata in standard form can be obtained. In this algorithm, we normalize the given automaton and obtain the normal nondeterministic finite automaton, which is unique in the equivalence class of automata. The sets of input sequences from any states of the normal automaton are disjoint. Therefore only by merging some states of the normal automaton, all the minimum equivalent automata can be obtained.

The number of the minimum nondeterministic finite automata in standard form equivalent to the given automaton is also discussed. We define the expanded reduced automaton matrix. Any reduced automaton matrices are included in the expanded matrix. Then, the number of the included matrix in the expanded matrix corresponds to the upper bound of the number of the standard formed minimum automata.

非決定性有限オートマトンの状態数最小化

仙石浩明

内容梗概

非決定性有限オートマトンは形式言語理論の根幹を成し、符号理論とも密接に関連している。また、様々な応用分野において重要な役割を果たしている。従って非決定性有限オートマトンの性質を調べ、理解することが重要である。本論文では、与えられたオートマトンに等価で状態数が最小の非決定性有限オートマトンについて考察し、最小化アルゴリズムを提案する。

決定性有限オートマトンの場合は、等価な状態を併合することにより唯一の既約な決定性有限オートマトンを作ることが出来る。さらに既約な決定性有限オートマトンは、等価な決定性有限オートマトンの中で最小の状態数を持つことが知られている。ところが非決定性有限オートマトンの場合は、一般に既約なものが状態数最小であるとは限らない。ある状態数以下の非決定性オートマトンの個数は高々有限であるので、その全てを探索すれば状態数最小の非決定性オートマトンを常に得ることが出来る。しかしこの方法は探索空間が膨大なため現実的ではない。

最初に全解探索でない最小化アルゴリズムが提案されたのは、 Kameda, Weiner によってである。このアルゴリズムでは与えられた非決定性オートマトンを正規化した Reduced Automaton Matrix を作り、この Matrix から状態数最小の非決定性オートマトンを生成するのであるが、生成されたオートマトンの中には与えられたオートマトンと等価でないものがある。従って等価でない場合は別の非決定性オートマトンを生成する必要がある。

本論文では、与えられた非決定性有限オートマトンに等価なオートマトンのみを構成するアルゴリズムを提案し、その正当性を証明する。このアルゴリズムでは、まず与えられたオートマトンに等価な標準非決定性有限オートマトンを作る事により正規化を行う。標準非決定性有限オートマトンは各状態からの受理入力列の集合が互いに共有部分を持たない。このため等価な状態数最小の非決定性有限オートマトンは、標準オートマトンの状態を併合する事により構成する事が出来る。

本論文では、さらに与えられたオートマトンに等価な状態数最小の非決定性オートマトンの個数についても考察する。まず、全ての Reduced Automaton Matrix を小行列として含む拡大 Reduced Automaton Matrix を定義する。すなわち与えられたオートマトンに等価な最小の非決定性オートマトンの状態数を m とすると、このオートマトンに対応する Reduced Automaton Matrix は、行及び列を適当に交換する事により、拡大 Reduced Automaton Matrix X_m に小行列として含まれる。この時、含まれる小行列の個数が、状態数最小の等価な非決定性オートマトンの個数の上界になる。

Minimization of Nondeterministic Finite Automata

Contents

1 Introduction

The study of finite automata is related to the various fields of computer science. The equivalence between finite state automata and regular expressions was presented by Kleene[2]. Since then, the theory of automata is closely related to the theory of formal languages. For example, some algorithms for constructing an automaton derived from a regular expression using derivatives were presented. One of them is the method of constructing a deterministic automaton due to Brzozowski[4], and another is of constructing a "small" size nondeterministic automaton due to Berry and Sethi[5].

The relationships between the coding theory and the automata are also studied by numerous investigators. For example, for a prefix code X, the minimum deterministic automaton recognizing X^* coincides with the minimum deterministic automaton representing X^* as the stabilizer of a state. However in the case of a code X, the minimum deterministic automaton representing X^* as the stabilizer of a set of states has, in general, fewer states because the automaton is a kind of a nondeterministic automaton. The construction and the uniqueness of the automaton are still open[6].

A nondeterministic automaton also plays an important role in the various applications such as compiler[7], formal verification, hardware description languages, and so on. In the study of

compiler the efficient automatic synthesis of the lexical analyzer which uses properties of nondeterministic automata is proposed[9]. In the formal verification, sequential machines or concurrent programs to be verified is frequently modelled by a nondeterministic automaton named the Kripke Structure. The automata can be also used to compare the power of temporal logics[8], and conversely, the equivalence of the states of the automaton can be defined using the temporal logics.

Hence the study of the properties of nondeterministic automata is very important in the connection to many areas of computer science. Therefore, many attempts have been made. However there still remain some difficult problems. One of the hard problems is the minimization of the nondeterministic automaton.

In this paper, we consider the minimum nondeterministic finite automata which are equivalent to the given automaton, and present an effective algorithm to minimize nondeterministic automata. In the case of deterministic finite automata, by merging equivalent states of the given deterministic automaton, the unique reduced deterministic automaton can be obtained. Furthermore, the reduced deterministic automaton is the minimum state equivalent deterministic automaton. However in the case of nondeterministic automata, reduced automaton is not always minimum.

Because of the finiteness of nondeterministic automata, we

can always get the minimum automaton with exhaustive search of all automata which are equivalent to the given automaton. But this method is not practical.

The first non-exhaustive search algorithm was proposed by Kameda and Weiner[3]. In their algorithm, they make two deterministic automata which are equivalent to the given automaton and to the reversed automaton of the given one, respectively. Next, they make a reduced automaton matrix whose rows and columns correspond to the states of these deterministic automata, respectively, that is the canonical form of the given automaton.

Next, they synthesize automata from the matrix, but as they pointed, all the nondeterministic automata synthesized from the matrix do not recognize the same input sequences recognized by a given automaton. Thus, on finding the minimum automaton, when synthesized automaton is not equivalent to the given one, they search another automaton. The condition under which a non-equivalent automaton is synthesized is unknown.

In this thesis, we present a more effective algorithm: synthesized automata are always equivalent to the given one. First, we make the unique "normal" nondeterministic automaton by using subset construction. Because of the uniqueness of the minimum deterministic automaton, this normal automaton is unique, too. Secondly, we synthesize a minimum automaton from this normal automaton.

We define the standard form of a nondeterministic automaton. Then, all the minimum automata in standard form are obtained by the algorithm. We also consider the question: how many minimum automata are equivalent to the given automaton.

In Section 2, we describe some basic definitions used in later discussion. In Section 3, the minimization method by Kameda and Weiner is introduced. In Section 4, we propose the new minimization method. In Section 5, we discuss the number of the minimum automata. And Section 6 is for conclusion.

2 Preliminaries

In this section basic definitions used in the following sections are presented.

In the following, we always use Σ as an input alphabet to automata.

Definition 1 (Nondeterministic Finite Automaton) A nondeterministic finite automaton $\mathcal{A}$ over alphabet Σ is a quadruple

$$\mathcal{A} = (S, \delta, S_0, F),$$

where

- S: the finite set of states,
- δ: the transition function, $\delta : S \times \Sigma \to 2^S$,
- S_0: the set of initial states, $S_0 \subset S$ and $S_0 \neq \phi$,
- F: the set of accepting states, $F \subset S$ and $F \neq \phi$.

λ-transition (i.e. a transition which can occur without no input) is not allowed. □

The domain of the transition function δ is extended from $S \times \Sigma$ to $2^S \times \Sigma$. That is, for $R \subset S$, $x \in \Sigma$,

$$\delta(R, x) = \bigcup_{s \in R} \delta(s, x).$$

Furthermore, for $\sigma = x_1 x_2 \cdots x_n$,

$$\delta(R, \sigma) = \delta(\delta(\ldots \delta(\delta(R, x_1), x_2), \ldots, x_{n-1}), x_n).$$

For null input sequence ϵ, $\delta(R, \epsilon) = R$.

We assume that there are no states which are not reachable from the initial states. That is, $\bigcup_{\sigma \in \Sigma^*} \delta(S_0, \sigma) = S$.

Definition 2 (Deterministic Finite Automaton) A deterministic finite automaton is a special case of a nondeterministic finite automaton. That is, a nondeterministic finite automaton $\mathcal{A} = (S, \delta, S_0, F)$ is called a deterministic finite automaton, if and only if,

- $|S_0| = 1$ (i.e. the cardinality of the set S_0 is equal to 1),
- for any $\sigma \in \Sigma^*$, $|\delta(S_0, \sigma)| \leq 1$.

□

For an nondeterministic automaton $\mathcal{A} = (S, \delta, S_0, F)$, the behavior of $\mathcal{A}$ (denoted by $\mathrm{bh}(\mathcal{A})$) is a set

$$\mathrm{bh}(\mathcal{A}) = \{\sigma \in \Sigma^* | \delta(S_0, \sigma) \cap F \neq \phi\}.$$

That is, the behavior of $\mathcal{A}$ is a set of input sequences which are accepted by an automaton $\mathcal{A}$.

In the same way, for $s_i \in S$, $S_i \subset S$, the behavior from s_i and from S_i are respectively as follows:

$$\mathrm{bh}(\mathcal{A}, s_i) = \{\sigma \in \Sigma^* | \delta(s_i, \sigma) \cap F \neq \phi\},$$

$$\mathrm{bh}(\mathcal{A}, S_i) = \{\sigma \in \Sigma^* | \delta(S_i, \sigma) \cap F \neq \phi\}.$$

The first parameter $\mathcal{A}$ may be omitted when it is not ambiguous. For an empty set of states, $\mathrm{bh}(\phi) = \phi$.

By using the behavior, equivalence of automata, states, and sets of states are defined.

Definition 3 (Equivalent Automata) Automata $\mathcal{A}, \mathcal{B}$ are equivalent, if and only if $\mathrm{bh}(\mathcal{A}) = \mathrm{bh}(\mathcal{B})$. □

Definition 4 (Equivalent States) States $s_i, s_j \in S$ (or sets of states $S_i, S_j \subset S$) are equivalent, if and only if $\mathrm{bh}(s_i) = \mathrm{bh}(s_j)$ (or $\mathrm{bh}(S_i) = \mathrm{bh}(S_j)$). □

We assume any two states of a given automaton are not equivalent states, because we can always obtain such an automaton by simply merging equivalent states into one.

We also assume, for all $s \in S$, $\mathrm{bh}(s) \neq \phi$, because we can always obtain such an automaton by removing states s such that $\mathrm{bh}(s) = \phi$.

For any input sequence $\sigma = x_1 x_2 \cdots x_n$, the reversed sequence $\overline{\sigma}$ is the sequence with its alphabets arranged in the reversed order. That is, $\overline{\sigma} = x_n x_{n-1} \cdots x_2 x_1$. The reversed automaton which accepts the reversed sequences is defined.

Definition 5 (Reversed Automaton) For a nondeterministic finite automaton $\mathcal{A} = (S, \delta, S_0, F)$, the reversed automaton of $\mathcal{A}$ is $\overline{\mathcal{A}} = (S, \overline{\delta}, F, S_0)$, where

$$\forall x \in \Sigma, \forall s_i, s_j \in S \; [s_i \in \overline{\delta}(s_j, x) \Leftrightarrow s_j \in \delta(s_i, x)].$$

Clearly, it follows that $\overline{\overline{\mathcal{A}}} = \mathcal{A}$. □

For a given nondeterministic automaton, the equivalent deterministic automaton can be constructed. This operation is called subset construction.

Definition 6 (Subset Construction) For a nondeterministic finite automaton $\mathcal{A} = (S, \delta, S_0, F)$, the deterministic finite automaton equivalent to $\mathcal{A}$ is $D(\mathcal{A}) = (S_P, \delta_P, P_0, F_P)$, where

- $S_P = \{\delta(S_0, \sigma) | \sigma \in \Sigma^*\} \backslash \{\phi\} = \{p_1, p_2, \ldots, p_m\}$,
- $\delta_P(p_i, x) = \begin{cases} \{\delta(p_i, x)\} & \text{if } \delta(p_i, x) \neq \phi \\ \phi & \text{otherwise} \end{cases} \qquad (p_i \in S_P)$,
- $P_0 = \{S_0\}$,
- $F_P = \{p \in S_P | p \cap F \neq \phi\}$.

Note: $p_i \in S_P$ is the subset of S. $\forall p_i \in S_P$, $\forall x \in \Sigma$, if $\delta(p_i, x) \neq \phi$, then $\exists p_j \in S_P$, $p_j = \delta(p_i, x)$. $\exists p_j \in S_P$, $p_j = S_0$. □

Since $\forall p_i \in S_P$, $\mathrm{bh}(\mathcal{D}(\mathcal{A}), p_i) = \mathrm{bh}(\mathcal{A}, p_i)$, if sets of states p_i, p_j of $\mathcal{A}$ are equivalent, corresponding states of $D(\mathcal{A})$ can be merged.

By merging the equivalent states of $D(\mathcal{A})$, we can obtain the unique reduced equivalent automaton which has minimum states.

Definitely diagnosability[1] is one of the important properties of sequential machines, when we consider about the diagnosis of these machines. We extend this property to a deterministic automaton.

Definition 7 (Definitely Diagnosable Automaton) A deterministic finite automaton $\mathcal{A} = (S, \delta, S_0, F)$ is definitely diagnosable, if

and only if, for any state $s \in S$, there exists an input sequence σ which satisfy the following condition.

$$\mathrm{bh}(\delta(s,\sigma)) \neq \phi \text{ and } \forall s' \neq s,\ \mathrm{bh}(\delta(s',\sigma)) = \phi$$

□

If we consider an input-output pair of a sequential machine as an input alphabet (of a deterministic finite automaton), a definitely diagnosable sequential machine can be regarded as a definitely diagnosable automaton, because long enough sequences of input-output pair (which correspond to the distinguishing sequence) σ satisfy the above condition.

Note: A sequential machine is definitely diagnosable of order μ, if and only if, every input sequence of length μ is a distinguishing sequence.

A matrix called a reduced automaton matrix derived from a nondeterministic automaton is defined[3]. For every equivalence class of automata, the matrix is unique up to permutation.

Definition 8 (Reduced Automaton Matrix) For a given nondeterministic finite automaton $\mathcal{A} = (S, \delta, S_0, F)$, $S = \{s_1, s_2, \ldots, s_l\}$, let

- $D(\mathcal{A}) = (S_M, \delta_M, M_0, F_M)$, $S_M = \{m_1, m_2, \ldots, m_p\}$,
- $D(\overline{\mathcal{A}}) = (S_N, \delta_N, N_0, F_N)$, $S_N = \{n_1, n_2, \ldots, n_q\}$.

Then a $p \times q$ matrix (a_{ij}) is defined, where

$$a_{ij} = \begin{cases} 1 & \text{if } \exists s_k \in S,\ s_k \in m_i \cap n_j \\ 0 & \text{otherwise.} \end{cases}$$

If there are the same row or column vectors in the matrix, merge them into one. □

Over an reduced automaton matrix, grid[3] is defined.

Definition 9 (Grid) Given a reduced automaton matrix, if all the entries at the intersection of a set of rows $\{m_{i_1}, m_{i_2}, \ldots, m_{i_a}\}$ and a set of columns $\{n_{j_1}, n_{j_2}, \ldots, n_{j_b}\}$ are 1's, then this set of rows and columns is called a grid represented as follows.

$$g = \{m_{i_1}, m_{i_2}, \ldots, m_{i_a}; n_{j_1}, n_{j_2}, \ldots, n_{j_b}\}$$

□

A set of grids forms the "cover with grids", if and only if every 1 in the reduced automaton matrix belongs to at least one grid in the set.

3 Conventional Method

In this section, the method for minimizing nondeterministic automaton, proposed by Kameda and Weiner[3] is summarized.

To synthesize the minimum nondeterministic automata, an inverse operation of the subset construction is defined.

Definition 10 (Subset Assignment) Let $\mathcal{M} = (S_M, \delta_M, M_0, F_M)$ be a deterministic finite automaton. The pair $< S, f >$ is called a subset assignment to $\mathcal{M}$ if S is a finite set and $f : S_M \rightarrow 2^S \backslash \{\phi\}$ is a function. Such an f is called a subset assignment function. □

A nondeterministic automaton is constructed from the subset assignment using the following intersection rule.

Definition 11 (Intersection Rule) Let $\mathcal{M} = (S_M, \delta_M, M_0, F_M)$ be a deterministic finite automaton, and let $< S, f >$ be a subset assignment to $\mathcal{M}$. Then $I(S, f, \mathcal{M})$ is the nondeterministic finite automaton (S, δ, S_0, F), where, for $\forall s \in S$, $\forall m_i \in S_M$, and $\forall x \in \Sigma$,

- $S_0 = f(m_0)$ $(m_0 \in M_0$. Since $|M_0| = 1$, m_0 is unique.),
- $s \in F \Leftrightarrow [s \in f(m_i) \Rightarrow m_i \in F_M]$,
- $s' \in \delta(s, x) \Leftrightarrow [s \in f(m_i) \Rightarrow s' \in f(\delta_M(m_i, x))]$.

$I(S, f, \mathcal{M})$ is called the nondeterministic finite automaton obtained by the intersection rule from $\mathcal{M}$. □

A subset assignment $< S, f >$ is derived from the cover with grids. That is, for the set of grids S, a subset assignment function is

$$f(m_i) = \{g \in S | m_i \in g\}.$$

The number of grids in the cover is equal to the number of states of the nondeterministic automaton constructed using the intersection rule. Therefore finding the minimum cover with grids, the minimum automaton is obtained.

But there is a problem: the synthesized automaton is not always equivalent to the given one. We must check the equality of these two automata. If not equivalent, we must search another automaton.

The reason why the non-equivalent automata are synthesized, is that the reduced automaton matrix has no information about the transition of the given automaton.

Definition 12 (Prime Grid) A grid g_1 contains another grid g_2, if and only if all 1 entries contained in g_2 are also contained in g_1.

A grid is called a prime grid, if and only if it cannot be contained in any other grids. □

If an equivalent automaton is synthesized from a cover with grids, the cover with only prime grids can be derived. The equivalent automaton synthesized from this cover has less or equal

states than the original automaton[3]. Hence, in finding the minimum automaton, we have only to consider prime grids over a reduced automaton matrix. However all of the minimum automata cannot be constructed from the cover with only prime grids.

4 Minimization using the Normal Automaton

In this section we present a new algorithm which can construct the minimum automata efficiently. For a given automaton, we make the canonical form of it, named the normal nondeterministic finite automaton. All the minimum automata constructed from this normal automaton are equivalent to the given automaton.

4.1 Normal Nondeterministic Finite Automaton

In the case of a deterministic automaton, the minimum deterministic automaton can be obtained by merging equivalent states. However in the case of a nondeterministic automaton, merging equivalent states is not sufficient in order to get the minimum automaton. Some automata have a state whose behavior (i.e. set of accepting input sequences from the state) is the union of the behavior of other states. In this case, that state must be split into states which are equivalent to other states so that they can be merged with respective states and the number of states decreases.

Suppose there are three states s_1, s_2, s_3, for example, and behavior of these states is $\mathrm{bh}(s_1) = \alpha, \mathrm{bh}(s_2) = \beta$, and $\mathrm{bh}(s_3) = \alpha \cup \beta$ respectively, where $\alpha \neq \beta$, and neither α nor β is not an empty set. Any two of these three states are not equivalent. If s_3 is split into two states, namely s_4, s_5 whose behavior is α, β respectively, then s_4 can be merged with s_1 and s_5 with s_2, because their behavior is the same. After merging, there are only two states s_1, s_2, thus

the number of states decreases by merging states after splitting of states.

An automaton is convenient for minimization, whose states are split up completely so that the behavior of any two states do not contain common input sequences. That is to say, behavior of all states is disjoint. We therefore define the following automaton.

Definition 13 (Disjoint Nondeterministic Finite Automaton) A nondeterministic finite automaton $\mathcal{A} = (S, \delta, S_0, F)$, is a disjoint nondeterministic finite automaton, if and only if, for all distinct states $s, s' \in S$, $\mathrm{bh}(s) \cap \mathrm{bh}(s') = \phi$. □

Lemma 1 A disjoint automaton has exactly one accepting state.

Proof: Suppose there are more than one accepting states. Let two of these states are s, s'. $\epsilon \in \mathrm{bh}(s)$ and $\epsilon \in \mathrm{bh}(s')$, thus $\mathrm{bh}(s) \cap \mathrm{bh}(s') \neq \phi$. This is the contradiction to the hypothesis. Q.E.D.

Theorem 1 (Disjoint Nondeterministic Finite Automaton) $\mathcal{A}$ is a disjoint nondeterministic finite automaton, if and only if $\overline{\mathcal{A}}$ is a deterministic automaton.

Proof: *[Necessary Condition]* Following from the Lemma 1, $\mathcal{A}$ has exactly one accepting state. Let this accepting state be s_F. For any state s of $\mathcal{A}$, $\mathrm{bh}(s)$ is disjoint. This implies, for any input sequence $\sigma \in \Sigma^*$, if $\overline{\sigma}$ is received by $\overline{\mathcal{A}}$ in state s_F, automaton $\overline{\mathcal{A}}$ transits to at most one state s, such that $\sigma \in \mathrm{bh}(\mathcal{A}, s)$. Thus $\overline{\mathcal{A}}$ is a deterministic automaton whose initial state is s_F.

[Sufficient Condition] Suppose there exist distinct states s, s' of $\mathcal{A}$ such that $\mathrm{bh}(s) \cap \mathrm{bh}(s') \neq \phi$. Let $\sigma \in \mathrm{bh}(s) \cap \mathrm{bh}(s')$. When $\overline{\sigma}$ is received by $\overline{\mathcal{A}}$ in the initial state, this automaton transits to states corresponding to s, s' simultaneously. This is the contradiction to the fact that $\overline{\mathcal{A}}$ is a deterministic automaton. Thus, for any two states s, s' of $\mathcal{A}$, $\mathrm{bh}(s) \cap \mathrm{bh}(s') = \phi$. Thus $\mathcal{A}$ is a disjoint automaton. Q.E.D.

Theorem 1 leads to the method of constructing a disjoint automaton equivalent to the given automaton.

Corollary: Given a nondeterministic finite automaton $\mathcal{A}$, make the deterministic automaton $\mathcal{B}$, which is equivalent to $\overline{\mathcal{A}}$, using subset construction. That is $\mathcal{B} = D(\overline{\mathcal{A}})$.

Then the reversed automaton of $\mathcal{B}$ is a disjoint automaton $\mathcal{D} = \overline{\mathcal{B}}$ which is equivalent to $\mathcal{A}$. □

Example: We construct a disjoint nondeterministic automaton equivalent to a nondeterministic automaton $\mathcal{A}$ shown in Figure 1. Note that "$\rightarrow$ 2" means the state "2" is an initial state, and "$\boxed{1}$" means the state "1" is an accepting state.

First, we make $\overline{\mathcal{A}}$, the reversed automaton of $\mathcal{A}$, shown in Figure 2. Secondly, we make a deterministic automaton equivalent to $\overline{\mathcal{A}}$, using subset construction. $D(\mathcal{A})$ is obtained shown in Figure 3. Finally, a disjoint nondeterministic automaton $\mathcal{D}$ shown in Figure 4 is obtained by making reversed automaton of $D(\overline{\mathcal{A}})$. □

Conversely, by merging some states of the disjoint automaton $\mathcal{D}$, the original automaton $\mathcal{A}$ is obtained.

$\mathcal{A}$	0	1
$\boxed{1}$	$1,2,3$	$-$
$\rightarrow 2$	$-$	$1,2$
$\rightarrow 3$	1	$2,3$

Figure 1: Nondeterministic Automaton $\mathcal{A}$

$\overline{\mathcal{A}}$	0	1
$\rightarrow 1$	$1,3$	2
$\boxed{2}$	1	$2,3$
$\boxed{3}$	1	3

Figure 2: Nondeterministic Automaton $\overline{\mathcal{A}}$

$D(\overline{\mathcal{A}})$	0	1
$\rightarrow \{1\}$	$\{1,3\}$	$\{2\}$
$\boxed{\{1,3\}}$	$\{1,3\}$	$\{2,3\}$
$\boxed{\{2\}}$	$\{1\}$	$\{2,3\}$
$\boxed{\{2,3\}}$	$\{1\}$	$\{2,3\}$

Figure 3: Deterministic Automaton $D(\overline{\mathcal{A}})$

$\mathcal{D}$	0	1
$\boxed{\{1\}}$	$\{2\}$, $\{2,3\}$	–
$\rightarrow \{1,3\}$	$\{1\}$, $\{1,3\}$	–
$\rightarrow \{2\}$	–	$\{1\}$
$\rightarrow \{2,3\}$	–	$\{1,3\}$, $\{2\}$, $\{2,3\}$

Figure 4: Disjoint Automaton $\mathcal{D} = \overline{D(\overline{\mathcal{A}})}$

Theorem 2 Let $D = (S_D, \delta_D, D_0, F_D)$ be the disjoint automaton constructed from the given automaton $\mathcal{A} = (S, \delta, S_0, F)$. Then,

$$\forall s \in S,\ \mathrm{bh}(\mathcal{A}, s) = \mathrm{bh}(\mathcal{D}, f(s)),$$

where $f : S \rightarrow 2^{S_D}$ is a function, such that

$$f(s) = \{d \in S_D | d \ni s\}.$$

Note: Since $\overline{\mathcal{D}}$ is the deterministic automaton constructed from $\overline{\mathcal{A}}$ by subset construction, a state of $\mathcal{D}$ is a subset of S.

Proof: Suppose $d \in f(s)$, that is, $s \in S$ is belongs to $d \in S_D$. Then, for any input sequence $\sigma \in \mathrm{bh}(\mathcal{D}, d)$,

$$d \in \overline{\delta_D}(F_D, \overline{\sigma}).$$

Following from Definition 10, it implies,

$$s \in \overline{\delta}(F, \overline{\sigma}).$$

Thus $\sigma \in \mathrm{bh}(\mathcal{A}, s)$. Therefore, $\mathrm{bh}(\mathcal{D}, d) \subset \mathrm{bh}(\mathcal{A}, s)$.

Conversely, for any input sequence $\sigma \in \mathrm{bh}(\mathcal{A}, s)$, it is obvious that,

$$\exists d \in f(s),\ \sigma \in \mathrm{bh}(\mathcal{D}, d).$$

Thus, $\mathrm{bh}(\mathcal{D}, f(s)) \supset \mathrm{bh}(\mathcal{A}, s)$. Therefore $\mathrm{bh}(\mathcal{A}, s) = \mathrm{bh}(\mathcal{D}, f(s))$. Q.E.D.

Example: We merge following states of a disjoint automaton $\mathcal{D}$ shown in Figure 4 and get states of the original automaton $\mathcal{A}$.

- merge states $\{1\}$, $\{1,3\}$ and get state 1.
- merge states $\{2\}$, $\{2,3\}$ and get state 2.
- merge states $\{1,3\}$, $\{2,3\}$ and get state 3.

□

A disjoint automaton is not unique. We therefore define the normal form of it.

Definition 14 (Normal Nondeterministic Finite Automaton) The minimum disjoint finite automaton among the equivalent disjoint automata, is the normal nondeterministic finite automaton. □

Lemma 2 Given a nondeterministic finite automaton $\mathcal{A} = (S, \delta, S_0, F)$, make the deterministic automaton which is equivalent to $\overline{\mathcal{A}}$ using subset construction. Minimize this automaton (i.e. $D(\overline{\mathcal{A}})$), and let it be $\mathcal{B}$.

Then the reversed automaton of $\mathcal{B}$ is the normal nondeterministic finite automaton $\mathcal{C} = \overline{\mathcal{B}}$ which is equivalent to $\mathcal{A}$.

Conversely, the reversed automaton of the normal automaton is the minimum deterministic automaton.

Proof: It follows from Theorem 1. Q.E.D.

Corollary: A normal automaton is unique in the equivalence class of automata.

Example: We construct the normal nondeterministic automaton equivalent to a nondeterministic automaton $\mathcal{A}$ shown in Figure 1.

Minimize a deterministic automaton $\mathcal{D}(\overline{\mathcal{A}})$ and get $\mathcal{B}$ shown in Figure 5 with renaming the states. That is,

- $s_1 = \{1\}$
- $s_2 = \{1, 3\}$
- merge equivalent states $\{2\}$, $\{2, 3\}$ and get s_3.

Then the normal nondeterministic automaton $\mathcal{C}$ shown in Figure 6 is obtained by making reversed automaton of $\mathcal{B}$. □

Since any state of the normal automaton is obtained by merging some equivalent states of the disjoint automaton, for any state of the disjoint automaton, a state of the normal automaton correspond.

We reconstructed the original automaton from the disjoint automaton. Then we consider the reconstruction from the normal automaton. That is, we merge some corresponding states of the normal automaton instead of the states of the disjoint automaton. Since the normal automaton is equivalent to the disjoint automaton, the automaton reconstructed from the normal automaton is also equivalent to the original automaton.

Definition 15 (Standard Form) A nondeterministic finite automaton $\mathcal{A}$ is in standard form, if and only if $D(\overline{\mathcal{A}})$ is the minimum deterministic automaton.

Since the disjoint automaton constructed from the standard formed automaton is the normal automaton, the reconstruction

$\mathcal{B}$	0	1
$\rightarrow s_1$	s_2	s_3
$\boxed{s_2}$	s_2	s_3
$\boxed{s_3}$	s_1	s_3

Figure 5: Minimized Deterministic Automaton $\mathcal{B}$

$\mathcal{C}$	0	1
$\boxed{s_1}$	s_3	$-$
$\rightarrow s_2$	s_1, s_2	$-$
$\rightarrow s_3$	$-$	s_1, s_2, s_3

Figure 6: Normal Automaton $\mathcal{C} = \overline{\mathcal{B}}$

from the normal automaton leads to the standard formed automaton.

Example: We merge following states of the normal automaton $\mathcal{C}$ shown in Figure 6 and get states of the standard formed automaton $\mathcal{A}'$ shown in Figure 7.

- merge states $s_1,\ s_2$ and get state 1.
- rename state s_3 to state 2.
- merge states $s_2,\ s_3$ and get state 3.

$\mathcal{A}'$ is obtained by adding a transition "$3 \xrightarrow{1} 1$" to $\mathcal{A}$. □

We can transform the nondeterministic automaton into its standard form by adding some extra transitions to the automaton. Therefore the number of states is unchangeable. Hence we only consider the standard formed nondeterministic finite automata in the following of this thesis.

All of the standard formed minimum nondeterministic automata equivalent to the normal automaton, can be obtained by merging some states of the normal nondeterministic automaton. We consider the way how to construct the minimum automaton from a given normal automaton.

Let a normal automaton be $\mathcal{N} = (S_N, \delta_N, N_0, F_N)$, where $S_N = \{n_1, n_2, \ldots, n_q\}$. The minimum automaton which is equivalent to $\mathcal{N}$ can be expressed as $\mathcal{A} = (S, \delta, S_0, F)$, where

- $S \subset 2^{S_N}$ i.e. $s_i \in S$ is a subset of S_N,

$\mathcal{A}'$	0	1
$\boxed{1}$	$1,2,3$	$-$
$\rightarrow 2$	$-$	$1,2$
$\rightarrow 3$	1	$1,2,3$

Figure 7: Standard Formed Nondeterministic Automaton $\mathcal{A}'$

- $\delta(s_i, x) = \mathrm{Map}(\delta_N(s_i, x))$,
- $S_0 = \mathrm{Map}(N_0)$,
- $F = \{s_i | s_i \cap F_N \neq \phi\}$.

$\mathrm{Map}(R_N)$ ($R_N \subset S_N$) is a subset of S that satisfy $\cup\mathrm{Map}(R_N) = R_N$. (Note: $\cup\mathrm{Map}(R_N)$ means the union of all sets $s_i \in \mathrm{Map}(R_N)$.) The set S, therefore, must be determined so that the function Map : $2^{S_N} \rightarrow 2^S$ can be defined.

Note, when we let $\mathrm{Map}(R_N) = \{s_i | s_i \subset R_N\}$, $\cup\mathrm{Map}(R_N) \subseteq R_N$ always holds. Thus it follows for any S, we can always define a function Map so that the set of $\mathcal{A}$'s accepting sequences $\mathrm{bh}(\mathcal{A})$ is included by the set of $\mathcal{N}$'s accepting sequences $\mathrm{bh}(\mathcal{N})$.

4.2 Minimization Algorithm

In this section, we present an algorithm to construct the nondeterministic finite automaton with the smallest number of states among automata equivalent to the given normal nondeterministic finite automaton.

Let $\mathcal{N} = (S_N, \delta_N, N_0, F_N)$ be the given normal automaton, and $\mathcal{A} = (S, \delta, S_0, F)$ be an automaton constructed from $\mathcal{N}$. All we have to do is to find the smallest $S \subset 2^{S_N}$, under which the function Map exists.

Theorem 3 For every transition $s_i \xrightarrow{x} s_j$ (i.e. $s_j \in \delta(s_i, x)$) of a nondeterministic finite automaton $\mathcal{A} = (S, \delta, S_0, F)$ which can be

constructed by merging some states of the normal nondeterministic finite automaton $\mathcal{N} = (S_N, \delta_N, N_0, F_N)$, the following condition holds.

$$\bigcup_{n_j \in s_j} \overline{\delta_N}(n_j, x) \subset s_i$$

Proof: Suppose there exists a transition $s_i \xrightarrow{x} s_j$ ($s_j \in \delta(s_i, x)$) where the above condition does not hold. Then, there exists $n_i \in S_N$ such that

$$n_i \in \bigcup_{n_j \in s_j} \overline{\delta_N}(n_j, x) \backslash s_i.$$

("\" is a difference set operator)

Since n_i is a state of the normal automaton, and $n_i \notin s_i$, thus $\mathrm{bh}(n_i) \cap \mathrm{bh}(s_i) = \phi$. On the other hand, $\delta_N(n_i, x) \subset s_j$, and $s_j \in \delta(s_i, x)$ which implies $s_j \subset \delta_N(s_i, x)$, thus

$$\mathrm{bh}(\delta_N(n_i, x)) \subset \mathrm{bh}(\delta_N(s_i, x)).$$

This is a contradiction to the fact $\mathrm{bh}(n_i) \cap \mathrm{bh}(s_i) = \phi$. Thus, there does not exists a transition where the theorem's condition does not hold. Q.E.D.

This theorem leads to the following theorem.

Theorem 4 If the reversed automaton of the normal automaton is definitely diagnosable, there does not exist an equivalent nondeterministic automaton which has fewer states than the normal automaton.

Note: the reversed automaton of the normal automaton is a deterministic automaton.

Proof: Suppose there exists an equivalent automaton $\mathcal{B}$ which has fewer states than the normal automaton $\mathcal{N}$. Let $\mathcal{A} = (S, \delta, S_0, F)$ be the reversed automaton of $\mathcal{B}$. Since $\mathcal{A}$ has fewer states than $\mathcal{N}$, $\mathcal{A}$ has a state s_i such that $s_i \supset \{n_i, n_j\}$ (n_i, n_j is a distinct states of $\mathcal{N}$.).

Following from Theorem 3, for all input alphabet $x \in \Sigma$, if $\mathcal{A}$ has a state $s_j \in \delta(s_i, x)$, then $s_j \supset \{\overline{\delta_N}(n_i, x), \overline{\delta_N}(n_j, x)\}$. Same things can be applied to s_j repeatedly. Hence for all input sequence $\sigma \in \Sigma^*$,

$$\overline{\delta_N}(n_i, \sigma) \neq \phi \Rightarrow \overline{\delta_N}(n_j, \sigma) \neq \phi.$$

It implies, for all input sequence $\sigma \in \Sigma^*$,

$$\mathrm{bh}(\overline{\delta_N}(n_i, \sigma)) \neq \phi \Rightarrow \mathrm{bh}(\delta(n_j, \sigma)) \neq \phi,$$

because all of the states of $\mathcal{B}$ are reachable. It violates the definitely diagnosability of $\overline{\mathcal{N}}$. Q.E.D.

Using the Theorem 3, search tree over $S \subset 2^{S_N}$ can be pruned. That is, to construct the minimum automaton, we add states one by one to S, which is initially an empty set, and if current S does not satisfy the condition of Theorem 3, we stop adding more states to S.

Example: A normal nondeterministic automaton $\mathcal{N} = (S_N, \delta_N, N_0, F_N)$, $S_N = \{0, 1, 2, 3, 4, 5\}$ is given as Figure 8.

First, let $S = \phi$, and we choose the elements of S (i.e. the states of the minimum automaton) one by one. As Figure 8 shows, $N_0 = \{2, 4, 5\}$, and the sets of the initial states of the normal au-

$\mathcal{N}$	0	1
$\boxed{0}$	–	–
1	0	4
$\rightarrow 2$	$1, 2, 3, 5$	$2, 3$
3	–	$0, 5$
$\rightarrow 4$	–	1
$\rightarrow 5$	4	–

Figure 8: Normal Automaton $\mathcal{N}$

tomaton and the minimum automaton must satisfy the equation: $S_0 = \mathrm{Map}(N_0)$. Thus S_0 is chosen from the following 5 possibilities.

1. $\{\{2,4,5\}\}$
2. $\{\{2,4\}, \{5\}\}$
3. $\{\{2,5\}, \{4\}\}$
4. $\{\{4,5\}, \{2\}\}$
5. $\{\{2\}, \{4\}, \{5\}\}$

We choose, for example, the first possibility. Let state $s_1 = \{2,4,5\}$, and $S_0 = \{s_1\}$. And add s_1 to S. Thus S becomes $\{s_1\}$.

Secondly, we consider the transitions from the state s_1. There are two transitions, namely $\delta(s_1, 0)$ and $\delta(s_1, 1)$. For $\delta(s_1, 0)$,

$$\delta(s_1, 0) = \mathrm{Map}(\delta_N(s_1, 0)), \qquad \delta_N(s_1, 0) = \{1,2,3,4,5\}.$$

Thus, we must add some states to S so that $\exists R \subset S$, $\cup R = \{1,2,3,4,5\}$ holds. Because $\{1,2,3,4,5\} \backslash s_1 = \{1,3\}$, states to be added are chosen from the following 2 possibilities.

- One state which contains both of $1, 3$.
- a state which contains 1 and a state which contains 3.

Notice these states can not contain 0. With 0 contained, s_1 would contain 0 because of Theorem 3 and $\overline{\delta_N}(0,0) = \{1\}$. This is the contradiction to the fact $s_1 = \{2,4,5\}$.

Definition 16 (Cover) $S \subset 2^{S_N}$, $R_{N1}, R_{N2} \subset S_N$ are given.

Cover(S, R_{N1}, R_{N2}) is defined to be the set consisting of all the sets R which satisfy following conditions.

- $R_{N1} \subset \cup R \subset S_N \backslash R_{N2}$
- $\forall s \in R,\ R_{N1} \not\subset \cup(R \backslash \{s\})$
- $\forall s \in R,\ \forall s' \in S,\ s \neq s'$

If $R_{N1} = \phi$, Cover$(S, \phi, R_{N2}) = \{\phi\}$. □

That is to say, Cover(S, R_{N1}, R_{N2}) is an enumeration of the possible sets of states to be added to S in order to cover R_{N1} and not to contain any elements of R_{N2}. Using this notation, for $\delta(s_1, 0)$, the set of states to be added to S can be written as $R \in$ Cover$(S, \{1, 3\}, \{0, 4, 5\})$. In the same way, for $\delta(s_1, 1)$, the set of states to be added to S can be chosen from Cover$(S, \{1, 2, 3\}, \{0, 4, 5\})$.

If it is not necessary to construct all of the minimum automata (i.e. in the case that it is sufficient only to construct one of the minimum automata), $R \in$ Cover(S, R_{N1}, R_{N2}), such that $\exists s \in R$, $\exists s' \in S$, $s \supset s'$, can be omitted, because $s'' = s \backslash s'$ can be used instead of s to cover R_{N1}. In other words, the third condition of the cover (i.e. $\forall s \in R$, $\forall s' \in S$, $s' \neq s$) can be replaced by the following condition.

- $\forall s \in R,\ \forall s' \in S,\ s \not\supset s'$

As $\{\{1, 2, 3\}\}$ is a member of both Cover$(S, \{1, 3\}, \{0, 4, 5\})$ and Cover$(S, \{1, 2, 3\}, \{0, 4, 5\})$, so we choose $\{\{1, 2, 3\}\}$. Let $s_2 = \{1, 2, 3\}$,

and add s_2 to S. Then, $\delta(s_1, 0) = \{s_1, s_2\}$, $\delta(s_1, 1) = \{s_2\}$. Now, $S = \{s_1, s_2\}$.

In the next phase, we consider the transition from s_2. The set of states to be added to S can be chosen from Cover$(S, \{0, 5\}, \{4\})$ and Cover$(S, \{0, 3\}, \{1\})$. As $\{\{0, 2, 3, 5\}\}$ is a member of both of them, let $s_3 = \{0, 2, 3, 5\}$ and add it to S. Then $S = \{s_1, s_2, s_3\}$.

Since $\delta(s_3, 0) = \{s_1, s_2\}$, $\delta(s_3, 1) = \{s_3\}$, no more states have to be added to S. Similarly searching other branches of possibilities, we'll find there are no solutions where the number of states is less than 3. Thus we obtain the minimum automaton shown in Figure 9. □

Minimization procedure goes as follows.

procedure Minimize($\mathcal{N} = (S_N, \delta_N, N_0, F_N)$)
    $m \leftarrow \infty$
    $S_{min} \leftarrow$**Undefined**
    for all $R \in$ Cover$(\phi, N_0, S_N \backslash N_0)$
        Search(R, R)
    return S_{min}
end

S_{min} is a variable for a set of states of the minimal automaton among the constructed automata. m is a variable such that $m = |S_{min}|$.

First, choose S_0 from Cover$(\phi, N_0, S_N \backslash N_0)$. That is, $\cup S_0$ must

	0	1
$\rightarrow s_1$	s_1, s_2	s_2
s_2	s_2, s_3	s_1, s_3
$\boxed{s_3}$	s_1, s_2	s_3

Figure 9: Minimum Nondeterministic Automaton

contain N_0 and may not contain anything else. Next, call Search procedure to search the transitions from S_0, and to assign the set of states of the minimum automaton to S_{min}. At the end of the procedure, return the minimum set of states.

Procedure Search(S, T) to search the transition from $T \subset S$ goes as follows. This procedure is called recursively, and pick up a state one by one from T to search the transitions from the state.

If $|S| \geq m$, no more search is needed, because no automata searched from S have less than m states. Thus return immediately. If $T = \phi$, all of the transitions are searched, that is, a new automaton is constructed. Since $|S| < m$, this automaton has fewer states than the old one. So let $S_{min} \leftarrow S$ and $m \leftarrow |S|$, and return.

The above process forms the former part of the Search procedure shown below.

Procedure Search(S, T)
    if $|S| \geq m$ **then return**
    if $T = \phi$ **then begin**
        $S_{min} \leftarrow S$
        $m \leftarrow |S|$
        return
    end

In this procedure, only one of the minimum automaton can be obtained, because when the minimum automaton is found, it imme-

diately returns. To get all of the minimum automata, the procedure must hold all of the sets of states of the minimum automata. We use the variable S_{min} to hold the set. If the automaton, such that $|S| < m$, is found, clear S_{min}, and let $m = |S|$. Then the former part of the Search procedure goes as follows.

Procedure Search(S, T)
    if $|S| > m$ **then return**
    if $T = \phi$ **then begin**
        if $|S| < m$ **then begin**
            $S_{min} \leftarrow \phi$
            $m \leftarrow |S|$
        end
        $S_{min} \leftarrow S_{min} \cup S$
        return
    end
    else if $|S| = m$ **then return**

The latter of the Search procedure goes as follows. Let s be one of the elements of T, remove s from T, and now search transitions from s. For each transition (i.e. $\forall x \in \Sigma$, $\delta(s, x)$), calculate the possible set of states to be added to S. Let R be the possible set.

Let R_N be the set of the normal automaton's states transited from s, which is covered with the states in $S \cup R$. The complementary set of R_N may not be covered with the states in R. Let

R' be the set of the normal automaton's states which is already covered by the states in S. Thus, the set covered by R is $R_N \backslash R'$. At the result, the possible sets of states to be added belong to the Cover$(S, R_N \backslash R', S_N \backslash R_N)$.

Then, for any possible set R, add R to S and search recursively, that is, call Search$(S \cup R, T \cup R)$.

$s \in T$
$T \leftarrow T \backslash \{s\}$
for all $x \in \Sigma$ **begin**
$R_N \leftarrow \bigcup_{n_i \in s} \delta_N(n_i, x)$
$R_{N2} \leftarrow S_N \backslash R_N$
$R' \leftarrow \{s_i \in S | s_i \subset R_N\}$
$R_{N1} \leftarrow R_N \backslash \cup R'$
for all $R \in$ Cover(S, R_{N1}, R_{N2})
Search$(S \cup R, T \cup R)$
end
end

4.3 Correctness of the Algorithm

In this section, we prove the correctness of the algorithm proposed in Section 4.2. First, we prove the property of finitely terminating. Secondly, we prove the soundness, that is, all the automata constructed by the algorithm are equivalent to the given normal

automaton. Finally, we prove the completeness, that is, all the equivalent minimum nondeterministic automata in standard form can be obtained.

Let $\mathcal{N} = (S_N, \delta_N, N_0, F_N)$ be the given normal nondeterministic finite automaton, which is an argument to the algorithm.

We begin with the property of finitely terminating.

Theorem 5 (Finitely Terminating) For any given automaton, the minimization algorithm terminates.

Proof: For any set $R \in \mathrm{Cover}(S, R_{N1}, R_{N2})$, $\forall s \in R$, $s \notin S$ (follows from Definition 16). It implies that all the elements added to T (added when Search($S \cup R$, $T \cup R$) is called) are different from each other. Since 2^{S_N} is a finite set, thus a set of all of the elements added to T is finite. Every time the procedure Search(S, T) is called, one element of T is removed by $T \leftarrow T \backslash \{s\}$. Hence T becomes an empty set in finite time.

Since the set Σ and the set $\mathrm{Cover}(S, R_{N1}, R_{N2})$ are finite, statements under the two "**for all**" statements in the Search(S, T) procedure repeat finitely.

Therefore the minimization procedure terminates finitely. Q.E.D.

Next, we prove the soundness of the algorithm.

Lemma 3 For the minimum nondeterministic finite automaton $\mathcal{A} = (S, \delta, S_0, F)$ constructed by the algorithm, the following conditions hold.

- $\forall s \in S$, $\delta(s, x) = \mathrm{Map}(\delta_N(s, x))$,
- $S_0 = \mathrm{Map}(N_0)$.

Proof: S_0 is chosen from $\mathrm{Cover}(\phi, N_0, S_N \backslash N_0)$. It implies $N_0 \subset \cup S_0 \subset S_N \backslash (S_N \backslash N_0)$. Thus $\cup S_0 = N_0$, that is, $S_0 = \mathrm{Map}(N_0)$.

Suppose there exist $s \in S$, $x \in \Sigma$, such that $\forall R \subset S$, $\delta_N(s, x) \neq \cup R$. Let $R_N = \delta_N(s, x)$. Following from the algorithm, there exists $S', R \subset S$, where

- $R_{N2} = S_N \backslash R_N$,
- $R' = \{s \in S' | s \subset R_N\}$,
- $R_{N1} = R_N \backslash \cup R'$,
- $\exists R \in S$, $R \in \mathrm{Cover}(S', R_{N1}, R_{N2})$.

Thus $R_N \backslash \cup R' \subset \cup R \subset R_N$ (follows from Definition 16). It implies $(\cup R) \cup (\cup R') = R_N$, that is $\cup (R \cup R') = R_N$, and $R \cup R' \subset S$. This is the contradiction. Q.E.D.

Theorem 6 (Soundness) All the automata constructed by the algorithm are equivalent to the given normal automaton.

Definition 17 For a integer l,

$$\mathrm{bh}_l(\mathcal{A}, s) = \{\sigma | \sigma \in \mathrm{bh}(\mathcal{A}, s) \wedge |\sigma| \leq l\},$$

where s may be a states or a set of states of an automaton $\mathcal{A}$. $\square$

Proof: Following from Lemma 3, for the minimum nondeterministic automaton $\mathcal{A} = (S, \delta, S_0, F)$ constructed by the algorithm, following conditions hold.

- $\forall s \in S,\ \delta(s, x) = \mathrm{Map}(\delta_N(s, x))$,
- $S_0 = \mathrm{Map}(N_0)$,
- $F = \{s | s \cap F_N \neq \phi\}$.

Suppose, for a integer l,

$$\forall s \in S,\ \mathrm{bh}_l(\mathcal{A}, s) = \mathrm{bh}_l(\mathcal{N}, s).$$

Since $\forall s \in S,\ \delta(s, x) = \mathrm{Map}(\delta_N(s, x))$,

$$\mathrm{bh}(\mathcal{A}, \delta(s, x)) = \mathrm{bh}(\mathcal{N}, \delta_N(s, x)).$$

Thus,

$$\forall s \in S,\ \mathrm{bh}_{l+1}(\mathcal{A}, s) = \mathrm{bh}_{l+1}(\mathcal{N}, s).$$

$\forall s \in S,\ \mathrm{bh}_0(\mathcal{A}, s) = \mathrm{bh}_0(\mathcal{N}, s)$, because

$$\begin{aligned} \mathrm{bh}_0(\mathcal{A}, s) &= \begin{cases} \epsilon & \text{if } s \cap F_N \neq \phi \\ \phi & \text{otherwise.} \end{cases} \\ &= \mathrm{bh}_0(\mathcal{N}, s). \end{aligned}$$

Hence, for any s, $\mathrm{bh}(\mathcal{A}, s) = \mathrm{bh}(\mathcal{N}, s)$. Since $S_0 = \mathrm{Map}(N_0)$, $\mathrm{bh}(\mathcal{A}) = \mathrm{bh}(\mathcal{N})$. Q.E.D.

Finally, we prove the completeness.

Theorem 7 (Completeness) All the standard formed minimum automata equivalent to the given automaton are constructed by the algorithm.

Proof: All the minimum automata in standard form are constructed by merging some states of the normal automaton.

Following from Definition 16, all the set of states of the normal automaton, which satisfy Theorem 3 can be chosen by the algorithm. Q.E.D.

5 The Number of the Minimum Automata

In this section, the number of the minimum automata in standard form equivalent to a given automaton is discussed.

5.1 Expanded Reduced Automaton Matrix

Let $\mathcal{A} = (S, \delta, S_0, F)$ $(S = \{s_1, s_2, \ldots, s_m\})$ be the minimum non-deterministic automaton. The deterministic automaton which is equivalent to $\mathcal{A}$ is $\mathcal{M} = D(\mathcal{A}) = (S_M, \delta_M, M_0, F_M)$. Then, S_M is a subset of the power set of S. We assume S_M is equal to the power set of S. Such an automaton is easily constructed by adding some transitions labeled with extra alphabets. For example, if a set $\{s_1, s_2\}$ does not appear in S_M, we add a new alphabet y to the set Σ and let $\delta(s_1, y) = \{s_1, s_2\}$. Then $D(\mathcal{A})$ transit from states which contain s_1 to the state which consists of s_1, s_2.

Similarly, we assume the set of states of $D(\overline{\mathcal{A}})$ is also equal to the power set of S. Then the reduced automaton of $\mathcal{A}$ is $(2^m - 1) \times (2^m - 1)$ matrix, where $m = |S|$. This matrix is covered by m grids. Since row vectors of this matrix are distinct each other, 1 entries of each row vector are covered by a distinct set of grids. The number of distinct sets from m elements is $2^m - 1$, thus it implies the set of row vectors are unique within permutation of the columns.

The same things can be applied to the column vectors, too. Consequently the reduced automaton matrix derived from this kind of automaton is unique, and only depends on the number of states

of the minimum automaton. We define this automaton as the expanded reduced automaton.

Definition 18 (Expanded Reduced Automaton Matrix) For given m, the $(2^m - 1) \times (2^m - 1)$ matrix, whose row or column vectors are distinct each other, covered with m grids is the expanded reduced automaton matrix X_m. □

Example: When $m = 3$, the expanded reduced automaton matrix goes as follows.

$$X_m = \begin{pmatrix} 1 & 0 & 1 & 0 & 1 & 0 & 1 \\ 0 & 1 & 1 & 0 & 0 & 1 & 1 \\ 1 & 1 & 1 & 0 & 1 & 1 & 1 \\ 0 & 0 & 0 & 1 & 1 & 1 & 1 \\ 1 & 0 & 1 & 1 & 1 & 1 & 1 \\ 0 & 1 & 1 & 1 & 1 & 1 & 1 \\ 1 & 1 & 1 & 1 & 1 & 1 & 1 \end{pmatrix}$$

This matrix is covered by following 3 grids.

- $g(X_m)_1 = \{1, 3, 5, 7; 1, 3, 5, 7\}$, i.e. a grid of 1,3,5,and 7th rows and columns.
- $g(X_m)_2 = \{2, 3, 6, 7; 2, 3, 6, 7\}$
- $g(X_m)_3 = \{4, 5, 6, 7; 4, 5, 6, 7\}$

□

Generally speaking, every expanded reduced automaton matrix X_m is covered by the following m grids after appropriate permutation of the rows and columns of the matrix.

$$g(X_m)_k = \{i | i\&(2^{k-1}) \neq 0;\ j | j\&(2^{k-1}) \neq 0\}$$

where $k = 1, 2, \ldots, m$, and "&" means the bitwise "and" operator.

5.2 The Number of the Minimum Automata

For any reduced automaton matrix which is covered with m grids, each row (or column) vector is covered with some of m grids. For every covering patterns from m grids (i.e. a set of grids which covers the all 1 entires of the vector), there exists the row (or column) of the expanded reduced automaton matrix X_m which has the same covering pattern, because every pattern exists in X_m. Thus, for any rows (or columns) of a reduced automaton matrix, the row (or column) of expanded matrix corresponds. It implies that any reduced automaton matrix which is covered with m grids is included in the X_m. In other word, by eliminating some rows and columns from X_m, every reduced automaton matrices are obtained.

Example: The reduced automaton matrix of the automaton $\mathcal{N}$ shown in Figure 8 goes as follows.

$$M = \begin{pmatrix} 0 & 0 & 1 & 0 & 1 & 1 \\ 0 & 1 & 1 & 1 & 1 & 1 \\ 1 & 1 & 1 & 1 & 1 & 1 \\ 0 & 1 & 1 & 1 & 0 & 0 \\ 1 & 1 & 1 & 1 & 0 & 1 \\ 1 & 0 & 1 & 1 & 1 & 1 \end{pmatrix}$$

This matrix is covered with following three grids, and each grid correspond to the state of the minimum automaton shown in Figure 9.

- $g(M)_1 = \{1, 2, 3, 6; 3, 5, 6\}$ (i.e. a grid of 1,2,3, and 6th rows and 3,5, and 6th columns) correspond to s_1.
- $g(M)_2 = \{2, 3, 4, 5; 2, 3, 4\}$ correspond to s_2.
- $g(M)_3 = \{3, 5, 6; 1, 3, 4, 6\}$ correspond to s_3.

After permutation of rows and columns of M: arrange 1,6,4,5,2, and 3rd row vector in this order, and 1,5,6,2,4,and 3rd column, following matrix M' is obtained.

$$M' = \begin{pmatrix} 0 & 1 & 1 & 0 & 0 & 1 \\ 1 & 1 & 1 & 0 & 1 & 1 \\ 0 & 0 & 0 & 1 & 1 & 1 \\ 1 & 0 & 1 & 1 & 1 & 1 \\ 0 & 1 & 1 & 1 & 1 & 1 \\ 1 & 1 & 1 & 1 & 1 & 1 \end{pmatrix}$$

M' is included in X_3 (i.e. M is the intersection of 2-7th rows and 1-5,7th columns of X_3).

Since the set of m grids which cover X_m is unique, determining which part of X_m a reduced automaton matrix is included, lead to the set of grids which cover the reduced automaton matrix. Further the set of grids lead to the minimum automaton.

Thus the number of the minimum automata, which is constructed from the normal automaton, is the number of reduced automaton matrices which are included in the expanded reduced automaton matrix. For a reduced automaton given (independent to an automaton itself), upper bound of the number of the minimum automata is the number of matrices included in the expanded matrix.

6 Conclusion

We have presented a minimization algorithm of nondeterministic finite automata. In the conventional method using the reduced automaton matrix proposed by Kameda and Weiner[3], the condition is unknown where the automaton, which is not equivalent to the given one, is synthesized from the matrix. We have made it clear by proposing the concept of the normal nondeterministic finite automaton. That is to say, the synthesized automaton is not equivalent where the condition of Theorem 3 does not hold.

We do not consider much about the efficiency of the proposed algorithm, because the algorithm as well as the conventional method depends on the subset construction whose complexity is already $O(2^n)$ where n is the number of states of the given automaton. The application of the minimization algorithm is a theme of future research.

The conventional method[3] can be combined with the proposed algorithm. That is, constructing the sets of states corresponding to the prime grid only instead of all the possible sets of states. It may be more efficient. However it cannot construct all of the minimum automaton, because the minimum automaton may have a state which does not correspond to the prime grid.

We have also presented a method to calculate the upper bound of the number of the minimum automata in standard form, using the expanded reduced automaton matrix. The upper bound

depends only to the reduced automaton matrix. In most cases, we conjecture, the number of the minimum automaton is 1. But the condition where the minimum automaton in standard form is unique is also left for the future work.

As we pointed, there are non-standard formed automata. Thus the number of the minimum automata may be larger. Moreover, given behavior of all the state, the function Map is not unique. Hence there are at least three stages in the varieties of the minimum nondeterministic automata. At present, we do not know how to handle such a vast variety of nondeterministic finite automata.

Acknowledgements

I would like to express my sincere appreciation to Professor Shuzo Yajima, who gave me the opportunity to study on logic design, for his continuous guidance, interesting suggestions, accurate criticisms and encouragements.

I would also like to express my thanks to Associate Professor Naofumi Takagi of Kyoto University who has been giving me invaluable suggestions.

I also wish to acknowledge Mr. Kiyoharu Hamaguchi and Mr. Yasuhiko Takenaga of Kyoto University for their helpful suggestions and valuable discussions. I am also grateful for their comments on the rough draft of this thesis. Thanks are also due to all the members of the Yajima Laboratory for useful discussions.

References

1. Z. Kohavi. *"Switching and Finite Automata Theory."* Tata McGraw-Hill, 1970

2. S. C. Kleene. *"Representation of Events in Nerve Nets and Finite Automata"* Automata Studies, Princeton University Press, 1956, pp.3-41

3. T. Kameda and P. Weiner. *"On the State Minimization of Nondeterministic Finite Automata"* IEEE Trans. Computer vol C-19, No. 7, July 1970

4. J. A. Brzozowski. *"Derivatives of Regular Expressions"* J. Assoc. Computing Machinery, vol. 11, pp.481-494, 1964

5. G. Berry and R. Sethi. *"From Regular Expressions to Deterministic Automata"* Theoretical Computer Science, vol. 48, 1986, pp.117-126

6. A. Restivo. *"Codes and Automata"* Lecture Notes in Computer Science vol 386, May 1988

7. A. V. Aho and J. D. Ullman. *"Principles of Compiler Design"* Addison-Wesley, 1977

8. M. C. Browne, E. M. Clarke, and O. Grümberg. *"Characterizing Kripke Structures in Temporal Logic"* CMU Tech. Report CS-87-104

9. M. Kato, Y. Inagaki, and N. Honda. *"An Efficient Algorithm for Synthesizing Finite Automata with Application to Generating Lexical Analyzer"* IEICEJ Tech. Report AL83-27

Appendix

List of Publications

- H. Sengoku and S. Yajima. *"Minimization of Nondeterministic Finite Automata"* Kokyuroku, Kyoto University Research Institute for Mathematical Science (to appear).

- H. Sengoku and A. Sugiyama. *"A Fast Convergence Algorithm for Adaptive FIR Filters with Simultaneous Adaptation of Coefficients and Tap Locations"* 1991 Spring National Convention Record of IEICEJ, Mar. 1991, A–205.

- A. Sugiyama and H. Sengoku. *"A Fast Convergence Algorithm for Adaptive FIR Filters with Simultaneous Adaptation of Coefficients and Tap Locations"* IEICEJ Tech. Report vol.90, No.466, CAS90-142, pp.53-60, Mar. 1991.