PAPER *Special Section on Foundations of Computer Science — New Trends of Theory of Computation and Algorithm —*

A New Finite Automata Construction Using a Prefix and a Suffix of Regular Expressions*

Hiroaki YAMAMOTO†a), ***Member and*** **Hiroshi FUJIWARA**†b), ***Nonmember***

SUMMARY This paper presents a new method to translate a regular expression into a nondeterministic finite automaton (an NFA for short). Let r be a regular expression and let M be a Thompson automaton for r. We first introduce a labeled Thompson automaton defined by assigning two types of expressions which denote prefixes and suffixes of words in $L(r)$ to each state of M. Then we give new ε-free NFAs constructed from a labeled Thompson automaton. These NFAs are called *a prefix equation automaton* and *a suffix equation automaton*. We show that a suffix equation automaton is isomorphic to an equation automaton defined by Antimirov. Furthermore we give an NFA called *a unified equation automaton* by joining two NFAs. Thus the number of states of a unified equation automaton can be smaller than that of an equation automaton.

key words: *regular expression, nondeterministic finite automaton*

1. Introduction

A regular expression (an RE for short) pattern matching problem plays an important role in the field of computer science, computational biology and so on. In general, RE pattern matching algorithms are designed by translating an RE into a nondeterministic finite automaton (NFA for short) or a deterministic finite automaton (DFA for short) [2], [13], [14], [17]. For this reason, developing an NFA or a DFA as compact as possible is important for designing efficient RE pattern matching algorithms, and has intensively been studied [1], [4]–[7], [9]–[11]. Furthermore compact automata are desired in a technique using bit-parallelism because this technique embeds an NFA or a DFA in computer-words. Bit-parallelism is known as a technique for developing faster matching algorithms. The aim of the paper is to present a new construction to translate an RE into a compact NFA.

Related works. *A Thompson automaton* (T-NFA for short) [16] is widely known as an NFA obtained from an RE, which is an NFA with ε-transitions and has at most $2m$ states and $4m$ transitions. T-NFAs can recursively be constructed based on the inductive definition of REs in $O(m)$ time and $O(m)$ space. *A position automaton* (PA for short, also called *a Glushkov automaton*) is also an NFA obtained from an RE, which is an ε-free NFA (that is, an NFA without any ε-transitions). A PA is constructed by focusing on occurrences of alphabet symbols in an RE. A PA has exactly $\tilde{m}+1$ states and at most $\tilde{m}^2+\tilde{m}$ transitions, and can be constructed in $O(\tilde{m}^2)$ time and $O(\tilde{m}^2)$ space [4], [8]. Here m is the total number of alphabet symbols and operators occurring in a given RE r and $\tilde{m}$ is the number of alphabet symbols occurring in r. Without loss of generality, we may assume $m = O(\tilde{m})$ as mentioned in [5].

As a more compact NFA, Antimirov [1] gave *an equation automaton* (EA for short) by introducing a notion of partial derivatives for an RE. An EA consists of at most $\tilde{m}+1$ states and at most $\tilde{m}^2+\tilde{m}$ transitions. Antimirov gave an algorithm generating an EA from an RE in $O(m^5)$ time. Champarnaud and Ziadi [6], [7] showed that an EA is obtained by merging equivalent states of an PA, and they gave an $O(m^2)$ time algorithm generating an EA. Thus an EA is a quotient of a PA. Khorsi, Ouardi and Ziadi [12] gave a more efficient algorithm using a minimization technique of a DFA. Their algorithm runs in $O(m \times State(EA))$ time, where $State(EA)$ is the number of states of the resulting EA. Ilie and Yu [11] gave another compact NFA called *a follow automaton*. A follow automaton is also a quotient of a PA and consists of at most $\tilde{m}+1$ states and at most $\tilde{m}^2+\tilde{m}$ transitions. They showed that a follow automaton and an EA are incomparable. Broda et al. [3] deeply discusses finite automata proposed until now and their dual version. Hromkovič et al. [10] presented a translation of an RE into an ε-free NFA with at most $2\tilde{m}$ states and $O(\tilde{m}(\log\tilde{m})^2)$ transitions. They reduce the number of transitions by increasing the number of states of a PA a little. Thus they improved the worst case of $O(m^2)$ transitions for the previously known ε-free NFA. In addition, they showed a lower bound $\Omega(\tilde{m}\log\tilde{m})$ on the number of transitions. We aim to reduce the number of states because it is easy to use bit-parallelism for simulating an NFA. In other words, if we can construct an NFA with small number of states, then we can pack all states of the NFA into a few computer words so that we can efficiently simulate an NFA using bit-parallelism.

Contributions of this paper. We will present a new technique to translate an RE into an NFA using an NFA called *a labeled Thompson automaton* (a labeled T-NFA for short). A labeled T-NFA is a T-NFA in which each state has two labels called a prefix label and a suffix label. Then we give new ε-free NFAs, *a prefix equation automaton (PreEA)* and *a suffix equation automaton (SufEA)*. We here make a remark on terminology. Broda et al. [3] called a PreEA *a*

Manuscript received April 10, 2020.
Manuscript revised August 14, 2020.
Manuscript publicized November 9, 2020.

†The authors are with the Faculty of Engineering, Shinshu University, Nagano-shi, 380–8553 Japan.

*A preliminary version of this paper appeared in the proceeding of NCMA 2014 [18].

a) E-mail: yamamoto@cs.shinshu-u.ac.jp (Corresponding author)
b) E-mail: fujiwara@cs.shinshu-u.ac.jp

DOI: 10.1587/transinf.2020FCP0010

prefix automaton and a SufEA *a suffix automaton*. However, we will use terminologies a PreEA and a SufEA because a suffix automaton has been used for an NFA constructed from all suffixes of a string. Now let r be an RE. To construct a labeled T-NFA, we introduce two labeling schemes, one is *a prefix-type labeling scheme* and the other is *a suffix-type labeling scheme*. A prefix-type labeling scheme assigns an RE denoting prefixes of words in $L(r)$ to a state q of a T-NFA, and a suffix-type labeling scheme assigns an RE denoting suffixes of words in $L(r)$, where $L(r)$ is a language denoted by r. One of the most significant advantages of a labeled T-NFA is that it enables us to join states using both expressions of prefixes and suffixes. We will show the following results.

- We present new NFAs obtained from a labeled T-NFA for any RE, that is a PreEA and a SufEA. Furthermore we show that a SufEA is isomorphic to an EA. A PreEA and a SufEA consist of at most $\tilde{m}+1$ states and at most $\tilde{m}^2 + \tilde{m}$ transitions.
- We give a more compact NFA called a unified equation automaton (UniEA) using both prefix labels and suffix labels. Since a SufEA is isomorphic to an EA, the number of states of a UniEA is smaller than or equal to that of an EA. Furthermore, by using an idea of a follow automaton [11], we can further reduce the number of states.

This paper is a revised version of [18], but the detail of a construction algorithm is omitted. In this paper, we focus on a construction method of a PreEA and a SufEA and give a complete proof for the result that a SufEA is isomorphic to an EA. In [18], an outline of the proof is only given. Furthermore we discuss an idea of a follow automaton in order to reduce the number of states. We can see the algorithm translating an RE into a PreEA, a SufEA, and a UniEA in $O(m^2)$ time and $O(m^2)$ space in [18].

The paper is organized as follows. In Sect. 2, we will give basic definitions of REs. In Sect. 3, we will give an outline of T-NFAs and EAs. In Sect. 4, we will introduce a labeled T-NFA, and present a PreEA and a SufEA. In Sect. 5 we will give a UniEA, and in Sect. 6 we introduce an idea of a follow automaton to reduce the number of states.

2. Regular Expressions and Some Notations

We here give some definitions for regular expressions.

Definition 1: Let Σ be an alphabet. The regular expressions over Σ are defined as follows.

1. $\emptyset$, ε (the empty word) and a ($\in \Sigma$) are REs that denote the empty set, the set $\{\varepsilon\}$ and the set $\{a\}$, respectively.
2. Let r_1 and r_2 be REs denoting the sets R_1 and R_2, respectively. Then $(r_1 + r_2)$, $(r_1 r_2)$ and (r_1^*) are also REs that denote the sets $R_1 \cup R_2$ (union), $R_1 R_2$ (concatenation), and R_1^* (Kleene closure or star), respectively.

Usually unnecessary parentheses in an RE are eliminated according to the following preference rules: Kleene closure has the higher preference than concatenation and union, and concatenation has the higher preference than union. We let $L(r)$ denote the language generated by an RE r. In this paper, the length of an RE denotes the number of alphabet symbols and operators (union, concatenation and Kleene closure) occurring in r. Furthermore, when m denotes the length of r, we let $\tilde{m}$ denote the number of alphabet symbols occurring in r. We will write r for $r\varepsilon$ and εr, and $\emptyset$ for $r\emptyset$ and $\emptyset r$ for any RE r because $r\varepsilon$ and εr denote a set $L(r)$, and $\emptyset r$ and $r\emptyset$ denote the empty set.

3. Thompson Automata and Equation Automata

3.1 Thompson Automata

A Thompson automaton (T-NFA) is recursively constructed from an RE according to the translation rules given in Fig. 1 (note that we can also see a similar construction in [9]). In Fig. 1, (a), (b) and (c) show the translations for $\emptyset$, ε and an alphabet symbol a, respectively, and (d), (e) and (f) show the translations for $(r_1 + r_2)$, $(r_1 r_2)$ and (r_1^*), respectively. Here M_1 and M_2 denote NFAs for r_1 and r_2, respectively.

Let $M = (Q, \Sigma, \delta, p_0, q_0)$ be a T-NFA for an RE r of length m, where Q is a set of states, Σ is an alphabet, δ is a transition function, p_0 is the initial state and q_0 is the final state. Note that a T-NFA has just one initial state and one final state. In addition, M consists of at most $2m$ states and $4m$ transitions. Furthermore, for any state $q \in Q$, all incoming transitions of q are caused either by the empty word ε or by an alphabet symbol $a \in \Sigma$. If all incoming transitions of q are caused by an alphabet symbol, then we call state q *a sym-state*; otherwise *an ε-state*. The following proposition holds for a T-NFA.

Proposition 1: For any RE r of length m, we can construct the T-NFA with at most $2m$ states and $4m$ transitions in $O(m)$ time and $O(m)$ space.

Example 1: Let us consider the RE $r = (a^*b + a^*ba + a^*)^*b$ over $\Sigma = \{a, b\}$. Figure 2 shows the T-NFA for r. The initial state is 0 and the final state is 25. Furthermore *sym*-states are states 5, 8, 11, 14, 16, 20 and 25.

Note that a position automaton is constructed from a

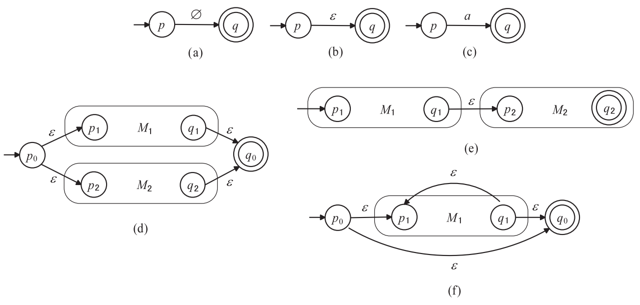

Fig. 1 Construction of a Thompson automaton: (a) $\emptyset$, (b) ε, (c) symbol $a \in \Sigma$, (d) union, (e) concatenation, (f) Kleene closure.

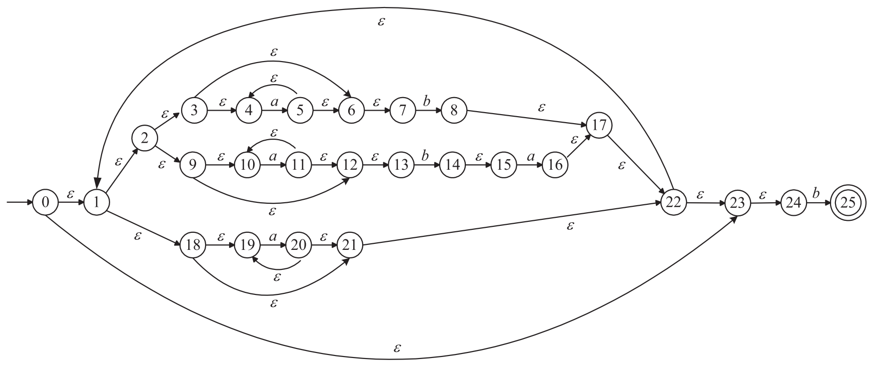

Fig. 2 Thompson automaton for $r = (a^*b + a^*ba + a^*)^*b$

T-NFA by eliminating ε transitions. In fact, a position automaton consists of the initial state and *sym*-states.

3.2 Equation Automata

We here show a definition of equation automata using partial derivatives of an RE by Antimirov [1]. Note that partial derivatives are also regular expressions. For any RE r and any $a \in \Sigma$, let us define the set $D_a(r)$ of partial derivatives according to the inductive definition of r below. In what follows, we define $D_a(r_1)r_2$ to be the set $\{\alpha_1 r_2, \dots, \alpha_t r_2\}$ for $D_a(r_1) = \{\alpha_1, \dots, \alpha_t\}$ and an RE r_2.

1. $D_a(\emptyset) = D_a(\varepsilon) = \emptyset$,
2. $D_a(a) = \{\varepsilon\}$ and $D_a(b) = \emptyset$ $(a \neq b, b \in \Sigma)$,
3. if $r = r_1 + r_2$ then $D_a(r) = D_a(r_1) \cup D_a(r_2)$,
4. if $r = r_1 r_2$ then
   a. if $\varepsilon \notin L(r_1)$ then $D_a(r) = D_a(r_1)r_2$;
   b. if $\varepsilon \in L(r_1)$ then $D_a(r) = D_a(r_1)r_2 \cup D_a(r_2)$,
5. if $r = r_1^*$ then $D_a(r) = D_a(r_1)r_1^*$.

We extend the above definition to any word over an alphabet Σ. Let $\mathcal{R}$ be a set of REs. For any $a \in \Sigma$, we define $D_a(\mathcal{R}) = \cup_{r \in \mathcal{R}} D_a(r)$. Then

1. for the empty word ε, $D_\varepsilon(\mathcal{R}) = \mathcal{R}$,
2. for any word $w \in \Sigma^*$ and any symbol $a \in \Sigma$, $D_{aw}(\mathcal{R}) = D_w(D_a(\mathcal{R}))$.

Now, let $PD(r)$ be the set $\{p \mid$ for some $w \in \Sigma^*$, $p \in D_w(r)\ \}$ of partial derivatives. Then an equation automaton (EA for short) $E = (Q, \Sigma, \delta, p_0, F)$ for r is defined as follows.

- $Q = PD(r)$,
- for any $p \in Q$ and symbol $a \in \Sigma$, $\delta(p, a) = D_a(p)$,
- $p_0 = r$, $F = \{p \mid \varepsilon \in L(p)\}$.

Example 2: We show partial derivatives and an equation automaton for $r = (a^*b + a^*ba + a^*)^*b$. Let us compute $PD(r)$. Here, for simplicity, $(a^*b + a^*ba + a^*)^*$ is represented by α. Hence $r = \alpha b$. Then partial derivatives are computed as follows:

$D_\varepsilon(r) = \{\alpha b\}$, $D_a(\alpha b) = \{a^*b\alpha b, a^*ba\alpha b, a^*\alpha b\}$,
$D_b(\alpha b) = \{\alpha b, a\alpha b, \varepsilon\}$,

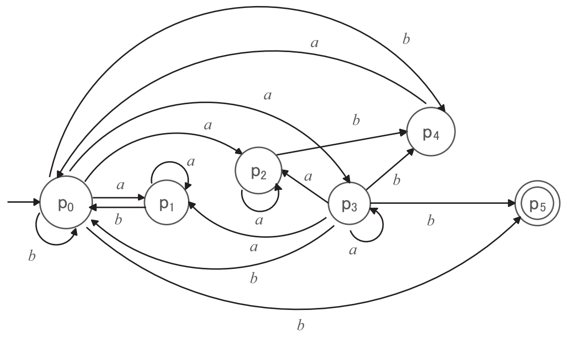

Fig. 3 The equation automaton for $r = (a^*b + a^*ba + a^*)^*b$

$D_a(a^*b\alpha b) = \{a^*b\alpha b\}$, $D_b(a^*b\alpha b) = \{\alpha b\}$,
$D_a(a^*ba\alpha b) = \{a^*ba\alpha b\}$, $D_b(a^*ba\alpha b) = \{a\alpha b\}$,
$D_a(a^*\alpha b) = \{a^*\alpha b, a^*b\alpha b, a^*ba\alpha b\}$, $D_b(a^*\alpha b) = \{\alpha b, a\alpha b\}$,
$D_a(a\alpha b) = \{\alpha b\}$, $D_b(a\alpha b) = \emptyset$.

Hence we have $PD(r) = \{\alpha b, a^*b\alpha b, a^*ba\alpha b, a^*\alpha b, a\alpha b, \varepsilon\}$. The equation automaton is given in Fig. 3. Here $p_0 = \alpha b$ (that is, r), $p_1 = a^*b\alpha b$, $p_2 = a^*ba\alpha b$, $p_3 = a^*\alpha b$, $p_4 = a\alpha b$ and $p_5 = \varepsilon$.

4. Prefix and Suffix Equation Automata

In this section, we give new automata called *a prefix equation automaton (PreEA)* and *a suffix equation automaton (SufEA)*. We first introduce a labeled T-NFA, and then show that a PreEA and a SufEA are obtained from a labeled T-NFA.

4.1 Labeled Thompson Automata

We introduce a labeled T-NFA $M = (Q, \Sigma, \delta, p_0, q_0, LP, LS)$. Here LP and LS are called *a prefix labeling function* and *a suffix labeling function*, respectively, which assign an RE to each state of Q. The functions LP and LS are defined by *a prefix-type labeling scheme* and *a suffix-type labeling scheme* given below. For each state q of M, $LP(q)$ and $LS(q)$ are REs, and $LP(q)$ denotes the set of all words leading from the initial state to state q and $LS(q)$ denotes the set of all words leading from state q to the final state of M. Let r be an RE on Σ. Then a prefix-type labeling scheme and a suffix-type labeling scheme are defined according to the construction given in Fig. 1.

[Prefix-type labeling scheme]

1. Basic labeling rules. Let p_0 be the initial state and q_0 be the final state for a T-NFA obtained from $\emptyset$, ε and $a \in \Sigma$:

   a. if $r = \emptyset$, then $LP(p_0) = \varepsilon$ and $LP(q_0) = \emptyset$,
   b. if $r = \varepsilon$, then $LP(p_0) = \varepsilon$ and $LP(q_0) = \varepsilon$,
   c. if $r = a$ ($a \in \Sigma$), $LP(p_0) = \varepsilon$ and $LP(q_0) = a$.

2. Inductive labeling rules. Let r_1 and r_2 be REs, and let $M_1 = (Q_1, \Sigma, \delta_1, p_1, q_1, LP_1, LS_1)$ and $M_2 = (Q_2, \Sigma, \delta_2, p_2, q_2, LP_2, LS_2)$ be labeled T-NFAs for r_1 and r_2, respectively. Furthermore let $M = (Q, \Sigma, \delta, p_0, q_0, LP, LS)$ be a labeled T-NFA obtained from M_1 and M_2 according to the construction in Fig. 1. Then LP is defined as follows.

   a. if $r = r_1 + r_2$, then $LP(p_0) = \varepsilon$ and $LP(q_0) = (LP_1(q_1) + LP_2(q_2))$. For any other state $q \in Q$, if $q \in Q_1$ then $LP(q) = LP_1(q)$; if $q \in Q_2$ then $LP(q) = LP_2(q)$,
   b. if $r = r_1 r_2$, then for any state $q \in Q$,

      i. if $q \in Q_1$ then $LP(q) = LP_1(q)$;
      ii. if $q \in Q_2$ then $LP(q) = (LP_1(q_1)LP_2(q))$,

   c. if $r = r_1^*$, then $LP(p_0) = \varepsilon$, $LP(q_0) = (LP_1(q_1))^*$, and for all other states $q \in Q$, $LP(q) = (LP_1(q_1))^* LP_1(q)$.

[Suffix-type labeling scheme]

1. Basic labeling rules. Let p_0 be the initial state and q_0 be the final state for a T-NFA obtained from $\emptyset$, ε and $a \in \Sigma$:

   a. if $r = \emptyset$, then $LS(p_0) = \emptyset$ and $LS(q_0) = \varepsilon$,
   b. if $r = \varepsilon$, then $LS(p_0) = \varepsilon$ and $LS(q_0) = \varepsilon$,
   c. if $r = a$ ($a \in \Sigma$), $LS(p_0) = a$ and $LS(q_0) = \varepsilon$.

2. Inductive labeling rules. Let r_1 and r_2 be REs, and let $M_1 = (Q_1, \Sigma, \delta_1, p_1, q_1, LP_1, LS_1)$ and $M_2 = (Q_2, \Sigma, \delta_2, p_2, q_2, LP_2, LS_2)$ be labeled T-NFAs for r_1 and r_2, respectively. Furthermore let $M = (Q, \Sigma, \delta, p_0, q_0, LP, LS)$ be a labeled T-NFA obtained from M_1 and M_2 according to the construction in Fig. 1. Then LS is defined as follows.

   a. if $r = r_1 + r_2$, then $LS(p_0) = (LS_1(p_1) + LS_2(p_2))$ and $LS(q_0) = \varepsilon$. For any other state $q \in Q$, if $q \in Q_1$ then $LS(q) = LS_1(q)$; if $q \in Q_2$ then $LS(q) = LS_2(q)$,
   b. if $r = r_1 r_2$, then for any state $q \in Q$,

      i. if $q \in Q_2$ then $LS(q) = LS_2(q)$;
      ii. if $q \in Q_1$ then $LS(q) = (LS_1(q)LS_2(p_2))$,

   c. if $r = r_1^*$, then $LS(q_0) = \varepsilon$, $LS(p_0) = (LS_1(p_1))^*$, and for all other states $q \in Q$, $LS(q) = LS_1(q)(LS_1(p_1))^*$.

Remark 1: Let $M = (Q, \Sigma, \delta, p_0, q_0, LP, LS)$ be a labeled T-NFA for an RE r and $\bar{M} = (Q, \Sigma, \bar{\delta}, q_0, p_0, \bar{LP}, \bar{LS})$ for the reversal of r. Since a T-NFA has a symmetric structure, we have $LP(q) = \bar{LS}(q)$ and $LS(q) = \bar{LP}(q)$. According to the labeling schemes, we can make labels $LP(q)$ and $LS(q)$ while constructing a T-NFA.

We simply call $LP(q)$ *a prefix label* and $LS(q)$ *a suffix label* for each state q. We have the following lemma for $LP(q)$ and $LS(q)$.

Lemma 1: Let M be the labeled T-NFA for a given RE r. Then, for any state q of M, $LP(q)$ is an RE which describes the set of words leading from the initial state of M to q, and $LS(q)$ is an RE which describes the set of words leading from q to the final state of M.

Proof : Let us prove the case $LP(q)$. The case $LS(q)$ can be proved similarly. The proof is by induction on an RE r. The base cases $r = \emptyset, \varepsilon$ and $a \in \Sigma$ are obvious. Assuming that the proposition holds for any REs r_1 and r_2, we will prove the cases $r_1 + r_2$, $r_1 r_2$ and r_1^*. The cases $r_1 + r_2$ and $r_1 r_2$ are obvious from the definition of $LP(q)$. Let us consider the case r_1^*. Since the initial state of M is labeled ε, the lemma holds for the initial state. The final state of M is labeled $(LP_1(q_1))^*$. This time, we note that $LP_1(q_1) = r_1$. Hence the final state satisfies the lemma. All other states q of M are just states of M_1, and are labeled $(LP_1(q_1))^* LP_1(q) = (r_1)^* LP_1(q)$. $LP_1(q)$ expresses an RE denoting the language consisting of words leading from the initial state to the state q of M_1. This implies that we can reach the state q by reading a word in $L(LP_1(q))$ after reading a word in $L(r_1^*)$ any number of times. Hence the lemma holds.

Example 3: We give an example of a labeled T-NFA. Let us consider the RE $r = (a^*b + a^*ba + a^*)^*b$ over $\{a, b\}$. Then the T-NFA M for r is given in Fig. 2 and Table 1 describes the prefix label $LP(q)$ and the suffix label $LS(q)$ of each state q of M.

4.2 PreEAs and SufEAs

Let us define PreEAs and SufEAs as quotients of a labeled T-NFA $M = (Q, \Sigma, \delta, p_0, q_0, LP, LS)$. Let $\tilde{Q}$ be the set consisting of all *sym*-states and the initial state p_0 of M. We define equivalence relations $\equiv_{pre}$ and $\equiv_{suf}$ over $\tilde{Q}$ as follows. For any state $p, q \in \tilde{Q}$, $p \equiv_{pre} q$ ($p \equiv_{suf} q$, resp.) if and only if $LP(p) \doteq LP(q)$ ($LS(p) \doteq LS(q)$, resp.), where $LP(p) \doteq LP(q)$ means that two expressions $LP(p)$ and $LP(q)$ are equal as a string in form without unnecessary parentheses. Furthermore, let $\tilde{Q}_{\equiv_{pre}}$ ($\tilde{Q}_{\equiv_{suf}}$, resp.) be the set which consists of all equivalence classes over $\tilde{Q}$ with respect to $\equiv_{pre}$ ($\equiv_{suf}$, resp.). Then we define a PreEA $PE = (Q_P, \Sigma, \delta_P, [p_0], F_P)$ as follows. Here we denote by $[q]$ an equivalence class for a state q.

Table 1 Labels of states of T-NFA given in Fig. 2. Note that $\alpha = (a^*b + a^*ba + a^*)^*$ and unnecessary parentheses are eliminated from labels.

state	prefix label	suffix label	state	prefix label	suffix label
0	ε	αb	13	αa^*	$ba\alpha b$
1	α	$(a^*b + a^*ba + a^*)\alpha b$	14	αa^*b	$a\alpha b$
2	α	$(a^*b + a^*ba)\alpha b$	15	αa^*b	$a\alpha b$
3	α	$a^*b\alpha b$	16	αa^*ba	αb
4	αa^*	$aa^*b\alpha b$	17	$\alpha(a^*b + a^*ba)$	αb
5	αa^*a	$a^*b\alpha b$	18	α	$a^*\alpha b$
6	αa^*	$b\alpha b$	19	αa^*	$aa^*\alpha b$
7	αa^*	$b\alpha b$	20	αa^*a	$a^*\alpha b$
8	αa^*b	αb	21	αa^*	αb
9	α	$a^*ba\alpha b$	22	$\alpha(a^*b + a^*ba + a^*)$	αb
10	αa^*	$aa^*ba\alpha b$	23	α	b
11	αa^*a	$a^*ba\alpha b$	24	α	b
12	αa^*	$ba\alpha b$	25	αb	ε

- $Q_P = \tilde{Q}_{\equiv_{pre}}$,
- for any $[p], [q] \in Q_P$ and symbol $a \in \Sigma$, $[q] \in \delta([p], a)$ if and only if there is a state $p' \in [p]$ and a state $q' \in [q]$ such that q' is reachable from p' by a symbol a,
- $[p_0]$ is the initial state (note that p_0 is the initial state of M),
- F_P is the set of final states and is defined as follows: for any $[p] \in Q_P$, $[p] \in F_P$ if and only if there is a state $p' \in [p]$ such that M can move from p' to the final state q_0 using only ε-transitions.

Similarly a SufEA $SE = (Q_S, \Sigma, \delta_S, [p_0], F_S)$ is defined by replacing Q_P by Q_S, $\tilde{Q}_{\equiv_{pre}}$ by $\tilde{Q}_{\equiv_{suf}}$, δ_P by δ_S and F_P by F_S.

Example 4: We give examples of a PreEA and a SufEA. As in Example 3, let us consider an RE $r = (a^*b + a^*ba + a^*)^*b$ over $\{a, b\}$. Then the T-NFA M for r is given in Fig. 2 and the labels of states of M are given in Table 1. The PreEA and the SufEA for r are constructed by using only *sym*-states and the initial state of M. Here the *sym*-states consists of $\{5, 8, 11, 14, 16, 20, 25\}$ and the initial state is 0. Now let us show a PreEA obtained from M. As seen in *prefix label* of Table 1, states 5, 11 and 20 are equivalent, 8 and 14 are also equivalent. Therefore we have the equivalence classes $\{0\}$, $\{5, 11, 20\}$, $\{8, 14\}$, $\{16\}$ and $\{25\}$. Thus we get the PreEA given in Fig. 4. Similarly, using *suffix label*, we get the equivalence classes $\{0, 8, 16\}$, $\{5\}$, $\{11\}$, $\{14\}$, $\{20\}$ and $\{25\}$. Hence we get the SufEA given in Fig. 5.

We have the following theorems. Theorem 1 follows from Lemma 1.

Theorem 1: Let r be an RE of length m, and let PE and SE be a PreEA and a SufEA obtained from r, respectively. Then PE and SE consist of at most $\tilde{m}+1$ states and $\tilde{m}(\tilde{m}+1)$ transitions, and accept $L(r)$.

Proof : It is clear to satisfy the size. Hence let us prove that a PreEA accepts $L(r)$. Let $M = (Q, \Sigma, \delta, p_0, q_0, LP, LS)$ be the labeled T-NFA for r and $PE = (Q_P, \Sigma, \delta_P, [p_0], F_P)$ be the PreEA obtained from M. Let $x \in \Sigma^*$ be any word accepted by M. Then there is a state $p_i \in \tilde{Q}$ such that M can reach p_i from p_0 by x and reach q_0 from p_i by ε-transitions. From Lemma 1, we have $x \in L(LP(p_i))$. Now let $[p_i] \in Q_P$.

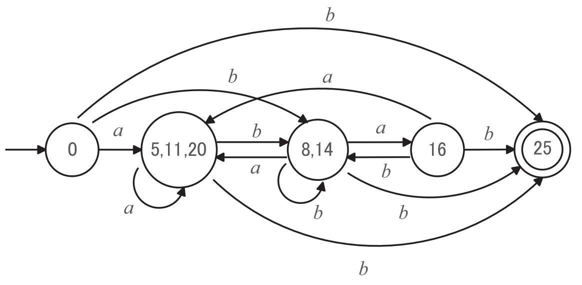

Fig. 4 The PreEA constructed from the T-NFA of Fig. 2.

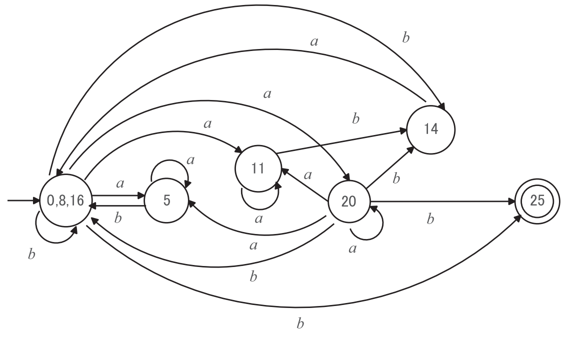

Fig. 5 The SufEA constructed from the T-NFA of Fig. 2.

Then the state $[p_i]$ is reachable from $[p_0]$ by x because $x \in L(LP(p_i))$. Furthermore $[p_i] \in F_P$ because M can reach q_0 from p_i by ε-transitions. Hence PE accepts x. Therefore if M accepts x then PE also accepts x. Similarly we can show that if PE accepts x, then M accepts x. Thus PE accepts $L(r)$. It is proved that SE accepts $L(r)$ in a similar way.

Let $M_1 = (Q_1, \Sigma, \delta_1, p_1, F_1)$ and $M_2 = (Q_2, \Sigma, \delta_2, p_2, F_2)$ be NFAs. It is said that M_1 is isomorphic to M_2 if there is an onto and one-to-one mapping f from Q_1 to Q_2 such that for any $a \in \Sigma$ and any $p, q \in Q_1$, a transition $q \in \delta_1(p, a)$ is defined if and only if a transition $f(q) \in \delta_2(f(p), a)$ is defined, $f(p_1) = p_2$, and $p \in F_1$ if and only $f(p) \in F_2$. We obtain the following theorem.

Theorem 2: Let r be an RE. Then, the SufEA is isomorphic to the EA for r.

Proof : The proof is by induction on an RE r. Let $M = (Q, \Sigma, \delta, p_0, q_0, LP, LS)$ be the labeled T-NFA obtained from r. Let $\tilde{Q}$ be the set of *sym*-states of M. We define $Suf(r)$ and $Suf^+(r)$ as follows:

- $Suf(r) = \{LS(q) \mid q \in \tilde{Q}$ and $LS(q) \neq \emptyset\}$,
- $Suf^+(r) = Suf(r) - \{LS(p_0)\}$.

Furthermore we define $PD^+(r) = \{\alpha \mid \exists w \in \Sigma^+, \alpha \in D_w(r)\}$. From the definitions of a SufEA and an EA, $Suf(r)$ denotes the set of REs corresponding to states of a SufEA, and $PD(r)$ denotes the set of states of an EA. Then the following claim holds. It follows from this claim that a SufEA is isomorphic to an EA. We consider an RE in which unnecessary parentheses are eliminated for $PD(r)$ and $Suf(r)$. Hence, for regular expressions r_1 and r_2, $r_1 = r_2$ means that r_1 and r_2 have the same form.

Claim : Let E be the EA and let N be the SufEA obtained from an RE r. Then there is an onto and one-to-one mapping f from $PD(r)$ to $Suf(r)$ such that

(1) for any $\alpha \in PD(r)$, $\alpha = f(\alpha)$, and
(2) for any $a \in \Sigma$ and any $\alpha_1, \alpha_2 \in PD(r)$, there is a transition of E from α_1 to α_2 by a if and only if a transition of N from $f(\alpha_1)$ to $f(\alpha_2)$ by a.

We can prove the claim according to the inductive construction of an RE. Clearly if there is an onto and one-to-one mapping from $PD(r)$ to $Suf(r)$ then there is an onto and one-to-one mapping from $PD^+(r)$ to $Suf^+(r)$ satisfying the conditions of the claim.

Basic case. $r = \emptyset, \varepsilon$, or $a \in \Sigma$.
We have $Suf(\emptyset) = PD(\emptyset) = \emptyset$, $Suf(\varepsilon) = PD(\varepsilon) = \{\varepsilon\}$, and $Suf(a) = PD(a) = \{a, \varepsilon\}$. It is obvious that we can define an onto and one-to-one mapping between $Suf(a)$ and $PD(a)$. Thus the claim holds.

Assume that the claim holds for REs with k or less operators. Let r be an RE with $k+1$ operators. Then we will prove three cases, (1) $r = r_1 r_2$, (2) $r = r_1^*$, and (3) $r = r_1 + r_2$.

Case 1. $r = r_1 r_2$.
We first compute $PD(r_1 r_2)$ according to the definition of partial derivatives. Suppose that $r_1 \neq \emptyset$ and $r_2 \neq \emptyset$. By the definition of partial derivatives, we have $PD(r_1 r_2) = \{\gamma \mid \exists w \in \Sigma^*, \gamma \in D_w(r_1 r_2)\}$. That is, $PD(r_1 r_2) = \cup_{w \in \Sigma^*} D_w(r_1 r_2)$. Hence we investigate REs constructing $D_w(r_1 r_2)$ for any word $w \in \Sigma^*$.

- $w = \varepsilon$: By the definition, $D_\varepsilon(r_1 r_2) = \{r_1 r_2\}$.
- $w = a_1 \cdots a_l$ ($l \geq 1$): In this case, if $\varepsilon \notin L(r_1)$ then $D_{a_1 \cdots a_l}(r_1 r_2) = D_{a_2 \cdots a_l}(D_{a_1}(r_1 r_2)) = D_{a_2 \cdots a_l}(D_{a_1}(r_1) r_2)$; if $\varepsilon \in L(r_1)$ then $D_{a_1 \cdots a_l}(r_1 r_2) = D_{a_2 \cdots a_l}(D_{a_1}(r_1) r_2 \cup D_{a_1}(r_2)) = D_{a_2 \cdots a_l}(D_{a_1}(r_1) r_2) \cup D_{a_2 \cdots a_l}(D_{a_1}(r_2))$. Let us compute iteratively $D_{a_2 \cdots a_l}$ $(D_{a_1}(r_1) r_2)$ according to the definition. Then we have $D_{w_2}(D_{w_1}(r_1) r_2) \subseteq D_{a_2 \cdots a_l}(D_{a_1}(r_1) r_2)$ for any w_1, w_2 such that $w = w_1 w_2$. Here we note that if $w' \in L(r_1)$ for any $w' \in \Sigma^*$, then there is an RE $\alpha \in D_{w'}(r_1)$ such that $\varepsilon \in L(\alpha)$. Let us consider any $w = w_1 w_2$ such that $w_1 \in L(r_1)$. Then $D_{w_1 w_2}(r_1 r_2)$ includes $D_{w_2}(D_{w_1}(r_1) r_2)$.
  Since $w_1 \in L(r_1)$, there exists $\alpha \in D_{w_1}(r_1)$ such that $\varepsilon \in L(\alpha)$. Hence $D_{w_2}(r_2) \subseteq D_{w_2}(D_{w_1}(r_1) r_2)$ because $D_{w_2}(\alpha r_2)$ includes $D_{w_2}(r_2)$. Thus, computing $D_w(r_1 r_2)$ for all words w over Σ^+, we have $\cup_{w \in \Sigma^+} D_w(r_1 r_2) = \{\cup_{w \in \Sigma^+} D_w(r_1) r_2\} \bigcup \{\cup_{w \in \Sigma^+} D_w(r_2)\} = PD^+(r_1) r_2 \cup PD^+(r_2)$.

From the above discussion, we have $PD(r_1 r_2) = PD^+(r_1) r_2 \cup PD^+(r_2) \cup \{r_1 r_2\} = PD(r_1) r_2 \cup PD^+(r_2)$. Next we evaluate $Suf(r)$. Let $M_1 = (Q_1, \Sigma, \delta_1, p_1, q_1, LP_1, LS_1)$ and $M_2 = (Q_2, \Sigma, \delta_2, p_2, q_2, LP_2, LS_2)$ be labeled T-NFAs for r_1 and r_2, respectively. Then $Suf(r_1 r_2) = \{LS_1(q) r_2 \mid q \in \tilde{Q}_1\} \cup \{LS_2(p) \mid p \in \tilde{Q}_2 - \{p_2\}\}$. Hence $Suf(r_1 r_2) = Suf(r_1) r_2 \cup Suf^+(r_2)$. By the induction assumption, there are onto and one-to-one mappings f_1 from $PD(r_1)$ to $Suf(r_1)$ and f_2 from $PD^+(r_2)$ to $Suf^+(r_2)$. Let us define a mapping f to be one such that for any $\alpha \in PD(r)$, if $\alpha \in PD(r_1) r_2$ then $f(\alpha) = f_1(\alpha_1) r_2$, where $\alpha = \alpha_1 r_2$; if $\alpha \in PD^+(r_2)$ then $f(\alpha) = f_2(\alpha)$. The mapping f is onto and one-to-one.

Let us show that f satisfies conditions (1) and (2) of the claim. Let E be the EA and N be the a SufEA for $r_1 r_2$. Note that $PD(r_1 r_2)$ is the set of states of E and $Suf(r_1 r_2)$ is the set of states of N. It is obvious that (1) holds because f_1 and f_2 satisfy the claim. Since $PD(r_1 r_2) = PD(r_1) r_2 \cup PD^+(r_2)$, if there a transition between α_1 and α_2 in $PD(r_1)$, then there a transition between $\alpha_1 r_2$ and $\alpha_2 r_2$ in $PD(r_1) r_2$ and vice versa. Similarly, this fact holds for $PD^+(r_2)$. Let us consider a transition of E from $PD(r_1) r_2$ to $PD^+(r_2)$. Here we note that there is no transitions from $PD^+(r_2)$ to $PD(r_1) r_2$. For any $\alpha \in PD(r_1) r_2$ and any symbol $a \in \Sigma$, assume that there is a transition by a from α to $\beta \in PD^+(r_2)$. Then we have $\beta \in D_a(\alpha)$ and can write $\alpha = \alpha_1 r_2$ where $\alpha_1 \in PD(r_1)$. Defining E_1 to be an EA for r_1, we have α_1 as a state of E_1. Since there is a transition from $\alpha_1 r_2$ to β, α_1 becomes a final state of E_1. Hence $\varepsilon \in L(\alpha_1)$ and then $D_a(\alpha_1 r_2) = D_a(\alpha_1) r_2 \cup D_a(r_2)$. Therefore $\beta \in D_a(r_2)$. Such an RE β is a label of a *sym*-state of M_2 reachable from the initial state by a. Thus N has a transition from α to β. Conversely there is a transition in E corresponding to that of N. Thus the mapping f satisfies the claim.

Case 2. $r = r_1^*$.
This case can be also proved in a similar way to that of $r_1 r_2$ as follows.

- $w = \varepsilon$: It follows from the definition that $D_\varepsilon(r_1^*) = \{r_1^*\}$.
- $w = a_1 \cdots a_l$ ($l \geq 1$). We have that $D_{a_1 \cdots a_l}(r_1^*) = D_{a_2 \cdots a_l}(D_{a_1}(r_1^*)) = D_{a_2 \cdots a_l}(D_{a_1}(r_1) r_1^*)$. Here $D_{a_1}(r_1) r_1^*$ consists of REs αr_1^* for any $\alpha \in D_{a_1}(r_1)$. By the definition of concatenation, if $\varepsilon \in L(\alpha)$ then $D_a(\alpha r_1^*) = D_a(\alpha) r_1^* \cup D_a(r_1^*)$ and if $\varepsilon \notin L(\alpha)$ then $D_a(\alpha r_1^*) = D_a(\alpha) r_1^*$. Let $w = w_1 w_2$ with $|w_1| \geq 1$. Then $D_w(r_1^*)$ has only partial derivatives of r_1^* obtained from $D_{w_2}(\alpha r_1^*)$ (some $\alpha \in D_{w_1}(r_1)$) and $D_{w_1}(r_1^*)$. Since w is any string over Σ^+, $\cup_{w \in \Sigma^+} D_w(r_1^*) = \cup_{w \in \Sigma^+} D_w(r_1) r_1^*$. Thus $PD^+(r_1^*) = \cup_{w \in \Sigma^+} D_w(r_1^*) = \cup_{w \in \Sigma^+} D_w(r_1) r_1^* = PD^+(r_1) r_1^*$.

From the above discussion, $PD(r_1^*) = \cup_{w \in \Sigma^*} D_w(r_1^*) = PD^+(r_1) r_1^* \cup \{r_1^*\}$. Next we compute $Suf(r_1^*)$. Let $M_1 =$

$(Q_1, \Sigma, \delta_1, p_1, q_1, LP_1, LS_1)$ be a labeled T-NFA for r_1. Then $Suf(r_1^*) = \{LS_1(q)r_1^* \mid q \in \tilde{Q}_1\}$. Hence since the initial state of M has a label r_1^*, $Suf(r_1^*) = Suf^+(r_1)r_1^* \cup \{r_1^*\}$. By the induction assumption, there is an onto and one-to-one mapping f_1 from $PD^+(r_1)$ to $Suf^+(r_1)$. Let us define a mapping f to be one such that for any $\alpha \in PD(r)$, if $\alpha \in PD^+(r_1)r_1^*$ then $f(r) = f_1(\alpha_1)r_1^*$; if $\alpha = r_1^*$ then $f(\alpha) = r_1^*$. In a similar way to concatenation, we have that f satisfies the claim.

Case 3. $r = r_1 + r_2$.
By the definition of $D_w(r)$, we have $PD(r_1+r_2) = PD^+(r_1) \cup PD^+(r_2) \cup \{r_1 + r_2\}$. On the other hand, $Suf(r_1 + r_2) = Suf^+(r_1) \cup Suf^+(r_2) \cup \{r_1 + r_2\}$. By the induction assumption, there are onto and one-to-one mappings f_1 from $PD^+(r_1)$ to $Suf^+(r_1)$ and f_2 from $PD^+(r_2)$ to $Suf^+(r_2)$. Let us define a mapping f to be one such that for any $\alpha \in PD(r)$, if $\alpha \in PD^+(r_1)$ then $f(\alpha) = f_1(\alpha)$; if $\alpha \in PD^+(r_2)$ then $f(\alpha) = f_2(\alpha)$; if $\alpha = r_1 + r_2$ then $f(\alpha) = r_1 + r_2$. This time, f satisfies the claim.

Thus the claim holds and then the theorem has been proved.

We can see that the SufEA in Fig. 5 is isomorphic to the EA in Fig. 3.

5. Unified Equation Automata

We give an algorithm to construct a unified equation automaton using a PreEA or a SufEA.

Step 1: We construct a labeled T-NFA $M = (Q, \Sigma, \delta, p_0, q_0, LP, LS)$ for a given RE, and compute two sets $\tilde{Q}_{\equiv_{pre}}$ and $\tilde{Q}_{\equiv_{suf}}$ of equivalence classes. If the number of classes of $\tilde{Q}_{\equiv_{pre}}$ is smaller than that of $\tilde{Q}_{\equiv_{suf}}$, then we construct a PreEA; otherwise a SufEA.

Step 2: Suppose that a PreEA M' was constructed in Step 1. Let $\tilde{Q}_{\equiv_{pre}}$ be the set of states of M'. We call an equivalence class consisting of just one state *a simple state* for $\tilde{Q}_{\equiv_{pre}}$. Then we get a UniEA by joining equivalent simple states of M' with respect to $\equiv_{suf}$.

Step 3: If a SufEA M' was constructed in Step 1, then we get a UniEA by joining equivalent simple states of M' with respect to $\equiv_{pre}$.

Since we choose a smaller set of $\tilde{Q}_{\equiv_{pre}}$ and $\tilde{Q}_{\equiv_{suf}}$ in Step 1, the number of states of a UniEA is always smaller than or equal to that of an EA. Assume that we chose $\tilde{Q}_{\equiv_{pre}}$ in Step 1. We join simple states with respect to $\equiv_{suf}$ in Step 2. Since a simple state is an equivalence class consisting of exactly one state, if suffix labels for two simple states are equal, we can join these simple states because the sets of words leading to final states from two states are same. This discussion does not hold for a non-simple state as follows. Suppose that $s_1 = \{q_1, q_2\}$ is a non-simple state and $s_2 = \{q_3\}$ is a simple state after Step 1. That is, $LP(q_1) = LP(q_2)$ and $LP(q_1) \neq LP(q_3)$. This time, if $LS(q_1) = LS(q_2) = LS(q_3)$, then we can join s_1 and s_2. However if $LS(q_1) \neq LS(q_2)$, then we cannot always join s_1 and s_2. Therefore, we join only a simple state. We have the next theorem.

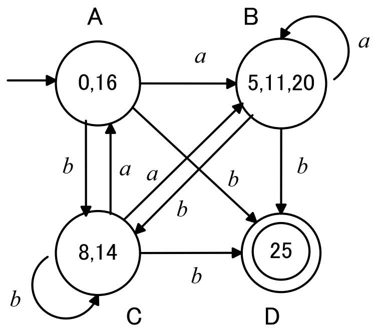

Fig. 6 The unified equation automaton obtained from Fig. 2.

Theorem 3: For any RE r of length m, let U be the UniEA obtained from r. Then U accepts $L(r)$. Furthermore the number of states of U is smaller than or equal to that of the EA obtained from r.

Let us give an example of a UniEA in Fig. 6. As you see in Fig. 4 and Fig. 5, the number of states of the PreEA is smaller than that of the SufEA. Hence we first construct the PreEA of Fig. 4. In the PreEA, states 0, 16 and 25 are simple. Since states 0 and 6 have the same suffix label, we can join these states. Finally we get the UniEA in Fig. 6. The following theorem is obtained from Theorem 1 and Theorem 2.

6. Further Reducing the Number of States

Using an idea of a follow automaton [11], we can reduce the number of states of a UniEA. A follow automaton is constructed from a position automaton by joining states with the same follow set. Since a position automaton is an NFA such that all the incoming transitions have the same symbol for any state, we define an equivalence relation by a follow set. Although a UniEA does not have such a property, we can introduce a similar equivalence relation as follows. Note that if follow sets are same for a position automaton, then the property (2) of Definition 2 holds. Therefore the equivalence relation by follow sets for a position is same as that of Definition 2. Nicaud [15] called a similar definition *strongly equivalent* for a DFA.

Definition 2: Let U be a UniEA. Then for any states p and q, p and q are equivalent if and only if the following two conditions holds: (1) both p and q or none are final, (2) for any $a \in \Sigma$, $\delta(p, a) = \delta(q, a)$.

If p and q are equivalent, then we can join two states p and q into one state. Hence it is clear that the following theorem holds.

Theorem 4: For any RE r of length m, let M be the NFA obtained from a UniEA by joining equivalent states. Then M accepts $L(r)$.

As an example of Theorem 4, Fig. 7 shows an NFA obtained from the UniEA of Fig. 6. First states A and B are joined and then states (A, B) and C are joined.

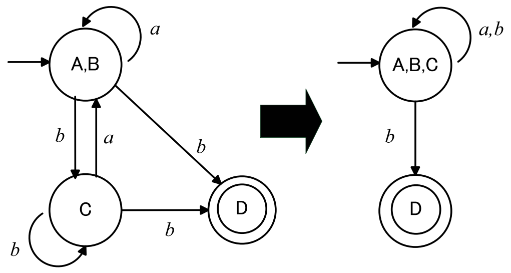

Fig. 7 Reducing the number of states of the unified equation automaton given in Fig. 6.

Remark 2: For any a regular language L, an NFA for the reverse of L is obtained from an NFA accepting L by reversing the transitions, and swapping the role of the initial and final states. Therefore we can apply Definition 2 to the reverse of a UniEA and then reverse the NFA again in order to obtain an NFA accepting the original language.

7. Concluding Remarks

We have shown a new method to construct an NFA from an RE. One of further researches is to design a more efficient algorithm than an algorithm given in [18].

Acknowledgments

The authors thank the reviewers for providing valuable comments to the paper. This work was supported by JSPS KAKENHI Grant Number JP17K00183.

References

[1] V. Antimirov, "Partial derivatives of regular expressions and finite automaton constructions," Theoret. Comput. Sci., vol.155, no.2, pp.291–319, March 1996.

[2] P. Bille, "New Algorithms for Regular Expression Matching," Proc. ICALP2006, vol.4501 of LNCS, pp.643–654, 2006.

[3] S. Broda, M. Holzer, E. Moaria, N. Moreira, and R. Reis, "A mesh of automata," Information and Computation, vol.265, pp.94–111, April 2019.

[4] A. Brüggemann-Klein, "Regular expressions into finite automata," Theoret. Comput. Sci., vol.120, no.2, pp.197–213, Nov. 1993.

[5] J.-M. Champarnaud, "Evaluation of three implicit structures to implement nondeterministic automata from regular expressions," Int. J. Found. Comput. Sci., vol.13, no.1, pp.99–113, 2002.

[6] J.-M. Champarnaud and D. Ziadi, From C-Continuations to New Quadratic Algorithms For Automaton Synthesis, J. Algebra Comput., vol.11, no.6, pp.707–735, 2001.

[7] J.-M. Champarnaud and D. Ziadi, "Canonical derivatives, partial derivatives and finite automaton construction, Theoret. Comput. Sci.," vol.289, no.1, pp.137–163, Oct. 2002.

[8] Chia-H. Chang and R. Paige, "From regular expressions to DFA's using compressed NFA's," Theoret. Comput. Sci., vol.178, no.1-2, pp.1–36, May 1997.

[9] J.E. Hopcroft and J.D. Ullman, Introduction to automata theory language and computation, Addison Wesley, 1979.

[10] J. Hromkovič, S. Seibert, and T. Wilke, "Translating Regular Expressions into Small ε-free Nondeterministic Finite Automata," J. Comput. Syst. Sci., vol.62, no.4, pp.565–588, June 2001.

[11] L. Ilie and S. Yu, "Follow Automata," Information and Computation, vol.186, no.1, pp.140–162, Oct. 2003.

[12] A. Khorsi, F. Ouardi, and D. Ziadi, "Fast equation automaton computation," J. Discrete Algorithms, vol.6, no.3, pp.433–448, Sept. 2008.

[13] G. Myers, "A Four Russians Algorithm for Regular Expression Pattern Matching," J. ACM, vol.39, no.4, pp.430–448, April 1992.

[14] G. Navarro and M. Raffinot, "New Techniques for Regular Expression Searching," Algorithmica, vol.41, no.2, pp.89–116, Sept. 2004.

[15] C. Nicaud, "Random Deterministic Automata," Proc. MFCS 2014, vol.8634 of LNCS, pp.5–23, 2014.

[16] K. Thompson, "Programming Techniques: Regular Expression Search Algorithm," Communications of the ACM, vol.11, no.6, pp.419–422, June 1968.

[17] H. Yamamoto, "Regular Expression Matching Algorithms using Dual Position Automata," JCMCC, vol.71, pp.103–125, 2009.

[18] H. Yamamoto, "A New Finite Automaton Construction for Regular Expressions," Proc. 6th NCMA, vol.304 of books@ocg.at, pp.249–264, 2014.

Hiroaki Yamamoto received the B.E. degree in information engineering from Shinshu University, Nagano, in 1980, and the M.E. and Ph.D. degrees in information engineering from Tohoku University, Sendai, in 1982 and 1985, respectively. From 1985 to 1988 he was with the Research Institute of Electrical Communication, Tohoku University, and with the Department of Information Engineering of Yamagata University. In 1988, he joined Shinshu University, Nagano, Japan, where he is currently a professor at the Department of Information Engineering. His research interests include pattern matching algorithms, automata theory, and information retrieval.

Hiroshi Fujiwara received his B.E., Master of Informatics, and Ph.D. in Informatics degrees from Kyoto University, in 2001, 2003, and 2006, respectively. Since 2014 he has been an associate professor in Department of Electrical and Computer Engineering at Shinshu University, Japan, after working at Kwansei Gakuin University and Toyohashi University of Technology. During his career, he worked as a research assistant at University of Freiburg, Germany, from 2004 to 2005, and worked as a visiting associate professor at University of Electronic Science and Technology, China, in 2012. His research interests include analysis of algorithms, online optimization, and functional analysis.