ISCAS-LCS-10-11 June 2010

中 国 科 学 院 软 件 研 究 所
计 算 机 科 学 实 验 室 报 告

Finite Automata of Expressions in the Case of Star Normal Form and One-Unambiguity

by

Haiming Chen

State Key Laboratory of Computer Science
Institute of Software
Chinese Academy of Sciences
Beijing 100190 China

Finite Automata of Expressions in the Case of Star Normal Form and One-Unambiguity ⋆

Haiming Chen

State Key Laboratory of Computer Science, Institute of Software,
Chinese Academy of Sciences, Beijing 100190, China
chm@ios.ac.cn

Abstract. Finite automata are basic for efficient implementation and application of regular expressions. Derivatives and partial derivatives are fundamental concepts for regular expressions which are useful tools to study automata construction from regular expressions. This paper studies derivatives, partial derivatives and automata in the case of expressions in star normal form as defined by Brüggemann-Klein, and of one-unambiguous expressions as defined by Brüggemann-Klein and Wood. If an expression is in star normal form, the paper first shows a property of derivatives which is stronger than Berry and Sethi's result. Second, it is known that the equation automaton and the follow automaton are two small automata, each of which is a quotient of the position automaton. For the relation between the equation and follow automata, however, Ilie and Yu stated that a rigorous analysis is necessary but difficult. The paper tackles the issue, and presents several results. For one-unambiguous expressions, the paper shows that the equation and follow automata can be computed in linear time, and sets up relations between the equation automaton and various constructions of Brzozowski automaton. In particular, when an expression is both one-unambiguous and in star normal form, we get a simple construction of the Brzozowski automaton, and show that the resulting automaton is equal to the equation automaton and that the original construction of Brzozowski automaton is a quotient of the equation automaton. Summary of relations of different sets of (partial) derivatives is presented.
keywords: Regular expressions, finite automata, derivatives, partial derivatives, star normal form, one-unambiguity

1 Introduction

Finite automata are basic for efficient implementation and application of regular expressions. Derivatives and partial derivatives are fundamental concepts for regular expressions which are useful tools to study automata construction from regular expressions. This paper studies derivatives, partial derivatives and automata in the case of regular expressions in star normal form, defined by Brüggemann-Klein [4], and of one-unambiguous expressions, defined by Brüggemann-Klein and Wood [6, 4, 5]. It is known that every regular expression can be turned into star normal form in linear time [4], and several algorithms depend on star normal form (e. g., [4, 8]). One-unambiguous expressions are often found in document definition languages. For example, in Document Type Definitions (DTDs) of XML the content models are required to be one-unambiguous expressions. A regular expression is one-unambiguous (or deterministic) if, informally, a symbol in the input word should be matched uniquely to a position in the regular expression without looking ahead in the word. It is known that the set of one-unambiguous languages is a proper subclass of regular languages. Since XML Schema both includes numeric occurrence indicators and requires one-unambiguity, one-unambiguous expressions with numeric occurrence indicators were studied [17]. Several authors investigated k-unambiguous regular expressions with a lookahead of k symbols [13, 14].

Derivatives of regular expressions were introduced by Brzozowski [7]. It has been established by Bruggemann-Klein and Wood [6] that one-unambiguous languages are closed under derivatives.

⋆ Work supported by the National Natural Science Foundation of China under Grants 60573013, 60721061.

The notion of derivatives was generalized to partial derivatives by Antimirov [1]. There has been no result about partial derivatives particular for one-unambiguous expressions. Among the many constructions of ϵ-free non-deterministic finite automata (NFA) from regular expressions, we consider *position automata* proposed separately by Glushkov [15] and McNaughton and Yamada [20], *equation automata* using partial derivatives [1], and *follow automata* proposed by Ilie and Yu [16]. The position automaton has size at most quadratic and can be computed in quadratic time [4, 11, 22]. Berry and Sethi [2] showed that the position automaton has a natural connection with the derivatives. It is known that a regular expression is one-unambiguous iff its position automaton is deterministic [6]. The equation automaton has also been proved to be equivalent to the automaton constructed from the prebase [21]. Champarnaud and Ziadi [9] proposed a quadratic algorithm for computing the equation automata which improved very much the original algorithm [1], and proved that the equation automaton is a quotient of the position automaton. Ilie and Yu [16] proposed a simplified proof of the result. Lombardy and Sakarovitch [19] gave another proof in the more general setting of expressions with multiplicity which applies to present Boolean case. Recently Ilie and Yu [16] introduced the follow automaton which can be computed in quadratic time, and proved that the follow automaton is a quotient of the position automaton. Champarnaud, Nicart and Ziadi presented another quadratic algorithm [8] for computing the follow automaton. The construction of the Brzozowski deterministic finite automaton (DFA) [7] uses derivatives. In [1] a similar construction using partial derivatives was presented. The work of the present paper is as follows.

The paper studies relations of different automata, in the case of star normal form expressions and of one-unambiguous expressions. In particular:

- The paper discusses the relation between the equation and follow automata. It has been known that both the equation and follow automata are quotients of the position automaton. The question is what is the relation between the first two automata. In [16] Ilie and Yu compared some examples and stated that the two automata "are incomparable", and that "a more rigorous comparison" between the automata "should be done" but "seems difficult". The paper tackles the issue, giving several conditions for the following relations between the two automata: one is a quotient of the other, the converse, and the two automata are isomorphic. Our work thus shows, for the first time, there are different conditions under which the relation of the two automata is different.

  In concrete, it first presents several simple characterizations, in terms of derivatives, of the above relations between the two automata.

  It then considers conditions in terms of the structure of expressions for the relations. To this end, we find conditions that are connected to the relations, and give several properties of the conditions.

  We show that for an expression in star normal form satisfying *CSE* condition (see Section 4) the equation automaton is a quotient of the follow automaton. Champarnaud, Ouardi and Ziadi [10] gave a related result concerning this issue. Their main theorem (Theorem 4, p.11) states for a normalized regular expression, the size of the equation automaton is smaller than the size of the follow automaton. Normalized regular expressions, however, constitute only a very restrictive and small subset of expressions satisfying *CSE* condition. For examples, none of the expressions in Example 4 are normalized regular expressions, while they all satisfy *CSE* condition.

  We further present conditions for some special situations, in which the two automata are isomorphic or the follow automaton is a quotient of the equation automaton.
- For one-unambiguous expressions, the aforementioned three NFA's become deterministic. What are the relations among these and other DFA's? The paper sets up relations between the equation automaton and various constructions of Brzozowski automaton. When an expression is both one-unambiguous and in star normal form, we get a simple construction of the Brzozowski automaton, and prove that the resulting automaton is equal to the equation automaton. This

equality also leads to the fact that the original construction of Brzozowski automaton is a quotient of the equation automaton.

The paper considers constructions of automata for the special expressions. For one-unambiguous expressions, it is known that the position automaton can be constructed in linear time. Could the efficiency of construction of the equation or follow automaton be improved? In the paper we show that they can also be computed in linear time for this class of expressions.

In the meantime, the paper presents several properties of derivatives and/or partial derivatives. If an expression is in star normal form, the paper shows that the derivatives of the marked expression (see Section 2 for the explanation of marked expression) with respect to words ending with a same letter are either $\emptyset$ or unique, a stronger property than Berry and Sethi's result [2] which establishes that in general the above derivatives are either $\emptyset$ or similar. This uniqueness of derivatives is of course an attractive property. For a one-unambiguous expression in star normal form, we present simple computation of derivatives and partial derivatives, which is of help in practice.

There have been several sets of derivatives and partial derivatives which are used in construction of different automata. Relations among the sets, under different conditions, are discussed. The results, though most of which are quite straightforward, are the basis for comparing related automata, and have not been studied all together in the literature.

Summary of relations of different sets of (partial) derivatives are presented in the paper. There has been no similar result in the literature on the subject that addresses constructions on star normal form expressions, to the best knowledge of the author. Also there has been no result about most of the above constructions for one-unambiguous expressions.

Section 2 introduces notations and notions required in the paper. Derivatives and partial derivatives are discussed in Section 3. Expressions in different cases including star normal form, one-unambiguity, and a combination of the first two cases are considered respectively in Sections 4, 5, and 6. Section 7 presents two algorithms for checking inclusion of one-unambiguous expressions. Section 8 gives concluding remarks.

2 Preliminaries

We assume the reader to be familiar with basic regular language and automata theory, e.g., from [23], so that we introduce here only some notations and notions used later in the paper.

2.1 Regular expressions and finite automata

Let Σ be an alphabet of symbols. The size of Σ is denoted by $|\Sigma|$. The empty word is denoted by ε. The set of all finite words over Σ is denoted by Σ^*. A regular expression over Σ is $\emptyset, \varepsilon$ or $a \in \Sigma$, or is obtained from these by applying the following rules finitely many times: for two regular expressions E_1 and E_2, the union $E_1 + E_2$, the concatenation E_1E_2, and the star E_1^* are regular expressions. For a regular expression E, the language specified by E is denoted by $L(E)$. Define $\lambda(E) = \varepsilon$ if $\varepsilon \in L(E)$ and $\emptyset$ otherwise. The size of E is denoted by $|E|$ and is the length of E when written in postfix (parentheses are not counted). The number of symbol occurrences in E, or the alphabetic width of E, is denoted by $\|E\|$. The symbols that occur in E, which is the smallest alphabet of E, is denoted by Σ_E.

We assume that the rules $E + \emptyset = \emptyset + E = E, E\emptyset = \emptyset E = \emptyset$, and $E\varepsilon = \varepsilon E = E$ (rules-$\emptyset\varepsilon$) hold in the paper.

One-unambiguous regular expressions are also called deterministic regular expressions, the name came from Brüggemann-Klein and Wood [6].

For a regular expression we can mark symbols with subscripts so that in the marked expression each marked symbol occurs only once. For example $(a_1 + b_2)^*a_3b_4(a_5 + b_6)$ is a marking of the expression $(a+b)^*ab(a+b)$. A marking of an expression E is denoted by $\overline{E}$. The same notation will

also be used for dropping of subscripts from the marked symbols: $\overline{\overline{E}} = E$. We extend the notation for words and automata in the obvious way. It will be clear from the context whether $\overline{\cdot}$ adds or drops subscripts.

Definition 1. *An expression E is one-unambiguous iff, for all words $uxv, uyw \in L(\overline{E})$ where $|x| = |y| = 1$, if $x \neq y$ then $\overline{x} \neq \overline{y}$. A regular language is one-unambiguous if it is denoted by some one-unambiguous expression.*

Besides one-unambiguity, there is also a notion of unambiguous regular expressions [3]. It is known that for each ambiguous regular expression there is an unambiguous regular expression that denotes the same language. On the other hand, expressions that are not one-unambiguous may not be defined by one-unambiguous expressions that denote the same languages. In other words, the set of one-unambiguous languages is a proper subclass of regular languages.

For an expression E over Σ, we define the following functions:

$$first(E) = \{a \mid aw \in L(E), a \in \Sigma, w \in \Sigma^*\}$$
$$last(E) = \{a \mid wa \in L(E), w \in \Sigma^*, a \in \Sigma\}$$
$$follow(E, a) = \{b \mid uabv \in L(E), u, v \in \Sigma^*, b \in \Sigma\}, \text{ for } a \in \Sigma$$

One can easily write equivalent inductive definitions of the above functions on E, which is omitted here.

Define $followlast(E) = \{b \mid vbw \in L(E), v \in L(E), v \neq \varepsilon, b \in \Sigma, w \in \Sigma^*\}$.

Definition 2. *An expression E is in star normal form (SNF) [4] if, for each starred subexpression H^* of E, $followlast(\overline{H}) \cap first(\overline{H}) = \emptyset$ and $\varepsilon \notin L(H)$.*

It is known that regular expressions can be transformed to SNF in linear time [4].

A finite automaton is a quintuple $M = (Q, \Sigma, \delta, q_0, F)$, where Q is a finite set of states, Σ is the alphabet, $\delta \subseteq Q \times \Sigma \times Q$ is the transition mapping, q_0 is the start state, and $F \subseteq Q$ is the set of accepting states. Denote the language accepted by the automaton M by $L(M)$.

Let $\equiv \subseteq Q \times Q$ be an equivalence relation. For $q \in Q$, $[q]_\equiv$ denotes the equivalence class of q w.r.t. $\equiv$ and, for $R \subseteq Q, R/_\equiv$ denotes the quotient set $R/_\equiv = \{[q]_\equiv \mid q \in R\}$. We say that $\equiv$ is right invariant w.r.t. M iff (1) $\equiv \subseteq (Q - F)^2 \cup F^2$ and (2) for any $p, q \in Q, a \in \Sigma$, if $p \equiv q$, then $\delta(p, a)/_\equiv = \delta(q, a)/_\equiv$.

If $\equiv$ is right invariant, the quotient automaton of M is $M/_\equiv = (Q/_\equiv, \Sigma, \delta_\equiv, [q_0]_\equiv, F/_\equiv)$, where $\delta_\equiv = \{([p]_\equiv, a, [q]_\equiv) \mid (p, a, q) \in \delta\}$. One can prove that $L(M/_\equiv) = L(M)$.

2.2 Position automata

The position automaton was introduced independently by Glushkov [15] and McNaughton and Yamada [20]. The *position automaton* of E is

$$M_{\text{pos}}(E) = (Q_{\text{pos}}, \Sigma, \delta_{\text{pos}}, q_E, F_{\text{pos}}),$$

where

1. $Q_{\text{pos}} = \Sigma_{\overline{E}} \cup \{q_E\}$, q_E is a new state not in $\Sigma_{\overline{E}}$
2. $\delta_{\text{pos}}(q_E, a) = \{x \mid x \in first(\overline{E}), \overline{x} = a\}$ for $a \in \Sigma$
3. $\delta_{\text{pos}}(x, a) = \{y \mid y \in follow(\overline{E}, x), \overline{y} = a\}$ for $x \in \Sigma_{\overline{E}}$ and $a \in \Sigma$
4. $F_{\text{pos}} = \begin{cases} last(\overline{E}) \cup \{q_E\}, & \text{if } \varepsilon \in L(E), \\ last(\overline{E}), & \text{otherwise} \end{cases}$

For further purpose we set $last_0(\overline{E})$ equal to $last(\overline{E})$ if $\varepsilon \notin L(E)$ and $last(\overline{E}) \cup \{q_E\}$ otherwise, and extend $follow(\overline{E}, q_E) = first(\overline{E})$.

Example 1. The position automaton $M_{\text{pos}}(E_1)$ for the regular expression $E_1 = (ab(c + \varepsilon))^*$ is shown in Figure 1(a).

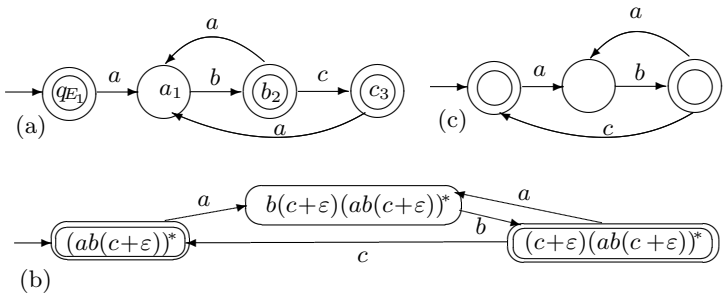

Fig. 1. (a) $M_{\mathrm{pos}}(E_1)$, (b) $M_{\mathrm{d_1}}(E_1)$ and $M_{\mathrm{pd}}(E_1)$, and (c) $M_{\mathrm{f}}(E_1)$, corresponding to $E_1 = (ab(c+\varepsilon))^*$.

As shown by Glushkov [15] and McNaughton and Yamada [20], $L(M_{\mathrm{pos}}(E)) = L(E)$.

$M_{\mathrm{pos}}(E)$ can be computed in quadratic time [4, 11, 22]. A quadratic time algorithm is given in [4], which has linear running time for one-unambiguous expressions. The algorithm uses the star normal form.

3 Derivatives

Derivatives of regular expressions were introduced by Brzozowski [7].

Definition 3. (Brzozowski [7]) *Given a regular expression E and a symbol a, the derivative $a^{-1}(E)$ of E w.r.t. a is defined inductively as follows:*

$$\begin{aligned}
a^{-1}(\emptyset) &= a^{-1}(\varepsilon) = \emptyset \\
a^{-1}(b) &= \begin{cases} \varepsilon, \text{ if } b = a \\ \emptyset, \text{ otherwise} \end{cases} \\
a^{-1}(F+G) &= a^{-1}(F) + a^{-1}(G) \\
a^{-1}(FG) &= \begin{cases} a^{-1}(F)G + a^{-1}(G), & \text{if } \varepsilon \in L(F) \\ a^{-1}(F)G, & \text{otherwise} \end{cases} \\
a^{-1}(F^*) &= a^{-1}(F)F^*
\end{aligned}$$

Derivative w.r.t. a word is computed by $\varepsilon^{-1}(E) = E$, $(wa)^{-1}(E) = a^{-1}(w^{-1}(E))$.

Partial derivatives were introduced by Antimirov [1].

Definition 4. (Antimirov [1]) *Given a regular expression E and a symbol a, the set of partial derivatives $\partial_a(E)$ of E w.r.t. a is defined as follows:*

$$\begin{aligned}
\partial_a(\emptyset) &= \partial_a(\varepsilon) = \emptyset \\
\partial_a(b) &= \begin{cases} \{\varepsilon\}, \text{ if } b = a \\ \emptyset, \quad \text{otherwise} \end{cases} \\
\partial_a(F+G) &= \partial_a(F) \cup \partial_a(G) \\
\partial_a(FG) &= \begin{cases} \partial_a(F)G \cup \partial_a(G), & \text{if } \varepsilon \in L(F) \\ \partial_a(F)G, & \text{otherwise} \end{cases} \\
\partial_a(F^*) &= \partial_a(F)F^*
\end{aligned}$$

Partial derivative w.r.t. a word is computed by $\partial_\varepsilon(E) = \{E\}$, $\partial_{wa}(E) = \bigcup_{p \in \partial_w(E)} \partial_a(p)$. The language denoted by $\partial_w(E)$ is $L(\partial_w(E)) = \bigcup_{p \in \partial_w(E)} L(p)$[1].

[1] $RF = \{EF | E \in R\}$ for a set R of regular expressions and a regular expression F.

Two regular expressions E_1 and E_2 which reduce to the same expression using associativity, commutativity, and idempotence of + are called *similar* [7], which is denoted $E_1 \sim_{aci} E_2$. The expressions E_1 and E_2 are equal, denoted $E_1 \equiv E_2$, if $L(E_1) = L(E_2)$. Let $D_1(E) = \{[w^{-1}(E)]_{\sim_{aci}} \mid w \in \Sigma^*\}$, $D_0 = \{[w^{-1}(E)]_{\equiv} \mid w \in \Sigma^*\}$, $DD(E) = \{\partial_w(E) \mid w \in \Sigma^*\}$, $PD(E) = \cup_{w\in\Sigma^*}\partial_w(E)$. These sets are useful in construction of automata from regular expressions. The equivalence of two sets A and B is denoted $A \sim B$ (i. e., there is a one-one correspondence between A and B). Let $[p]_{\sim_{aci}}, [q]_{\sim_{aci}} \in D_1(E)$, define $[p]_{\sim_{aci}} \approx [q]_{\sim_{aci}}$ if $p \equiv q$. We have the following properties.

Proposition 1. *For a regular expression E, we have*
(1) $D_0(E), D_1(E)$ are finite ([7]), and $D_0(E) \sim D_1(E)/_{\approx}$,
(2) $|PD(E)| \leq \|E\| + 1$ ([1]),
(3) $|D_0(E)| \leq |DD(E)| \leq 2^{\|E\|+1}$.

Proof. Below is the proof of (3). For a set $R = \{t_1, \ldots, t_k\}$ of regular expressions, denote ΣR an expression $t_1 + \ldots + t_k$ up to an arbitrary permutation. For any expression E and $w \in \Sigma^*$, $L(w^{-1}(E)) = L(\Sigma\partial_w(E))$ [1]. If $L({w_1}^{-1}(E)) = L({w_2}^{-1}(E))$, then $L(\Sigma\partial_{w_1}(E)) = L(\Sigma\partial_{w_2}(E))$, but not necessarily $\Sigma\partial_{w_1}(E) = \Sigma\partial_{w_2}(E)$, which implies $|D_0(E)| \leq |DD(E)|$. By definition, the number of elements in $DD(E)$ can not be more than $2^{\|E\|+1}$. □

The notion of derivatives leads to a very natural construction of the *Brzozowski deterministic automaton* [7], defined as

$$M_{\mathrm{d_1}}(E) = (D_1(E), \Sigma, \delta, E, \{p \in D_1(E) \mid \varepsilon \in L(p)\}),$$

where $\delta(p, a) = a^{-1}(p)$, for any $p \in D_1(E), a \in \Sigma$.

A similar construction was introduced in [1]:

$$M_{\mathrm{dd}}(E) = (DD(E), \Sigma, \delta, \{E\}, \{P \in DD(E) \mid \exists p{\in}P, \varepsilon \in L(p)\}),$$

where $\delta(P, a) {=} \cup_{p\in P}\partial_a(p)$, for any $P {\in} DD(E), a {\in} \Sigma$.

Example 2. $M_{\mathrm{d_1}}(E_1)$ for the expression E_1 from Example 1 is shown in Figure 1(b). $M_{\mathrm{dd}}(E_1)$ is the same as $M_{\mathrm{d_1}}(E_1)$.

The *equation automaton* [1] constructed by partial derivatives is

$$M_{\mathrm{pd}}(E) = (PD(E), \Sigma, \delta_{\mathrm{pd}}, E, \{q \in PD(E) \mid \varepsilon \in L(q)\}),$$

where $\delta_{\mathrm{pd}}(q, a) = \partial_a(q)$, for any $q \in PD(E), a \in \Sigma$. An example is shown in Figure 1(b).

It is proved that for a regular expression, the equation automaton is a quotient of the position automaton [9, 16]. Another proof is given by Lombardy and Sakarovitch [19], which is in the more general setting of expressions with multiplicity but still applies to present case (multiplicities over the Boolean semiring).

In general $M_{\mathrm{pd}}(E)$ is incomparable with $M_{\mathrm{d_1}}(E)$ or $M_{\mathrm{dd}}(E)$, but in some situations they are comparable; See Section 5,6.

Expressions with distinct symbols are called linear. For any expression E, $\overline{E}$ is the linearized version of E.

For linear expressions from Brzozowski [7] and Berry and Sethi [2] the following fact is easily derived.

Proposition 2. *Let E be linear. Given $x \in \Sigma_E$, for all words w,*
1. If $E = E_1 + E_2$, then

$$(wx)^{-1}(E_1 + E_2) = \begin{cases} (wx)^{-1}(E_1) \text{ if } x \in \Sigma_{E_1} \\ (wx)^{-1}(E_2) \text{ if } x \in \Sigma_{E_2} \end{cases} \quad (1)$$

2. If $E = E_1E_2$, then

$$(wx)^{-1}(E_1E_2) = \begin{cases} (wx)^{-1}(E_1)E_2 & \text{if } x \in \Sigma_{E_1} \\ (vx)^{-1}(E_2) & \text{if } w = uv, \varepsilon \in L(u^{-1}(E_1)), x \in \Sigma_{E_2}, \\ & u \in \Sigma_{E_1}^*, v \in \Sigma_{E_2}^* \\ \emptyset & \text{otherwise} \end{cases} \tag{2}$$

Proof. 1. It is directly from Berry and Sethi [2].

2. From Berry and Sethi [2] it is already known

$$(wx)^{-1}(E_1E_2) = \begin{cases} (wx)^{-1}(E_1)E_2 & \text{if } x \in \Sigma_{E_1} \text{ (a)} \\ \Sigma_{w=uv}\lambda(u^{-1}(E_1))(vx)^{-1}(E_2) & \text{otherwise (b)} \end{cases}$$

Let us consider (b) and set $wx = a_1a_2 \dots a_t$. For a concrete sequence of $a_1 \dots a_t$, a subterm $(a_{r+1} \dots a_t)^{-1}(E_2)$ in (b) can exist only if $a_1, \dots, a_r \in E_1$ and $a_{r+1}, \dots, a_t \in E_2$. Since $a_n, 1 \leq n \leq t$ is either in E_1 or in E_2, there is at most one such subterm in (b). If such condition is not satisfied, then $(wx)^{-1}(E_1E_2) = \emptyset$. □

4 Expressions in SNF

4.1 Derivatives

Berry and Sethi [2] have shown that, for a marked expression $\overline{E}$, given a fixed $x \in \Sigma_{\overline{E}}$, $(wx)^{-1}(\overline{E})$ is either $\emptyset$ or unique modulo $\sim_{aci}$ for all words w. In [2], based on this a natural connection between the position automaton and derivatives is set up.

Here, we further show that, if E is in SNF then the aci-similarity in the above is unnecessary.

Proposition 3. *For a marked expression $\overline{E}$, if E is in SNF then given a fixed $x \in \Sigma_{\overline{E}}$, $(wx)^{-1}(\overline{E})$ is either $\emptyset$ or unique for all words w.*

Proof. We prove it by induction on the structure of $\overline{E}$. The cases for $\overline{E} = \varepsilon, \emptyset, x$, $x \in \Sigma_{\overline{E}}$, are obvious.

1. $\overline{E} = \overline{E_1} + \overline{E_2}$. By Eq (1), if x is in $\overline{E_1}$, then $(wx)^{-1}(\overline{E_1})$ is left, and the inductive hypothesis applies to it. The same is for x in $\overline{E_2}$.

2. $\overline{E} = \overline{E_1E_2}$. If x is in $\overline{E_1}$, then by Eq (2) $(wx)^{-1}(\overline{E}) = (wx)^{-1}(\overline{E_1})\overline{E_2}$, and the inductive hypothesis applies to it. Otherwise, x is in $\overline{E_2}$ and $(wx)^{-1}(\overline{E}) = (vx)^{-1}(E_2)$ for some $w = uv$ or $(wx)^{-1}(\overline{E}) = \emptyset$. Therefore the inductive hypothesis applies to it.

3. $\overline{E} = \overline{E_1}^*$. From [7] and [2] $(wx)^{-1}(\overline{E})$ is a sum of subterms of the form $(vx)^{-1}(\overline{E_1})\overline{E_1}^*$ where $wx = uvx$. We show that there is at most one non-null subterm.

Suppose there are two non-null subterms $(v_1x)^{-1}(\overline{E_1})\overline{E_1}^*$ and $(v_2x)^{-1}(\overline{E_1})\overline{E_1}^*$. If $v_1 \neq v_2$, suppose $|v_1| < |v_2|$. Let $wx = a_1a_2 \dots a_t$. We can suppose $v_1x = a_{r_1} \dots a_t, v_2x = a_{r_2} \dots a_{r_1} \dots a_t, 1 \leq r_2 < r_1 \leq t$. Since $(v_1x)^{-1}(\overline{E_1}) \neq \emptyset$, we have $a_{r_1} \in first(\overline{E_1})$. Since $(v_2x)^{-1}(\overline{E_1}) \neq \emptyset$, there exists a word w_1, such that $a_{r_2} \dots a_{r_1} \dots a_tw_1 \in L(\overline{E_1})$. Then $a_{r_1} \in follow(\overline{E_1}, a_{r_1-1})$.

A careful analysis on the derivation of $(wx)^{-1}(\overline{E})$ shows that if $(v_1x)^{-1}(\overline{E_1}) \neq \emptyset$, then either $\varepsilon \in L((a_{r_1-1})^{-1}(\overline{E_1}))$ or $\varepsilon \in L((a_n \dots a_{r_1-1})^{-1}(\overline{E_1}))$ for some $n < a_{r_1-1}$. In either case, we have $a_{r_1-1} \in last(\overline{E_1})$. Therefore E is not in SNF, which is a contradiction.

If $v_1 = v_2$, then $v_1x = v_2x = a_{r_1} \dots a_t, 2 < r_1 \leq t$. Similarly, a careful analysis on $(wx)^{-1}(\overline{E})$ shows that there must be $\varepsilon \in L((a_{n_1} \dots a_i)^{-1}(\overline{E_1})), \varepsilon \in L((a_{n_2} \dots a_i)^{-1}(\overline{E_1}))$ and $\varepsilon \in L((a_{n_3} \dots a_{n_1-1})^{-1}(\overline{E_1})), n_2 < n_1 \leq i \leq r_1-1, n_3 < n_1$. So we have $a_{n_1} \in first(\overline{E_1}), a_{n_1} \in follow(\overline{E_1}, a_{n_1-1})$, $a_{n_1-1} \in last(\overline{E_1})$. Therefore E is not in SNF, which is a contradiction.

So there is at most one non-null subterm, and the inductive hypothesis applies to it. □

Corollary 1. *If E is in SNF and there are non-null $(w_1)^{-1}(\overline{E})$ and $(w_2)^{-1}(\overline{E})$, such that $(w_1)^{-1}(\overline{E}) \sim_{aci} (w_2)^{-1}(\overline{E})$, then $(w_1)^{-1}(\overline{E}) = (w_2)^{-1}(\overline{E})$.*

From the proof of Proposition 3 above, it follows

Corollary 2. *If $E = E_1^*$ is in SNF, then for a non-null $(wx)^{-1}(\overline{E})$, $(wx)^{-1}(\overline{E}) = (vx)^{-1}(\overline{E_1})\overline{E}$ for some $wx = uvx$.*

4.2 Equation and follow automata

This subsection discusses the relation between equation and follow automata.

The *follow automaton* $M_{\rm f}(E)$ was introduced by Ilie and Yu [16]. It is constructed by eliminating ε-transitions from an ε-automaton defined in [16]. We do not present the construction in detail here. Instead, an example is shown in Figure 1(c). What is important here is the following.

Define the equivalence $\equiv_f \subseteq Q_{\rm pos}^2$ by $x_1 \equiv_f x_2$ iff $x_1 \in last_0(\overline{E}) \Leftrightarrow x_2 \in last_0(\overline{E})$ and $follow(\overline{E}, x_1) = follow(\overline{E}, x_2)$. The equivalence relation is right invariant w.r.t. $M_{\rm pos}(E)$. Define $M_1 \simeq M_2$ if M_1 and M_2 are isomorphic. It is known that

Proposition 4. *$M_{\rm f}(E) \simeq M_{\rm pos}(E)/_{\equiv_f}$ ([16]).*

As we have mentioned, it is well-known that the equation automaton is a quotient of the position automaton [9, 16, 19]. Here it is presented following [16]. For a letter $x \in \Sigma_{\overline{E}}$, denote $C_x(\overline{E})$ any expression $(wx)^{-1}(\overline{E}) \neq \emptyset$. Denote also $C_{q_E}(\overline{E}) = \overline{E}$ (q_E is the start state of the position automaton of E). For a SNF expression E (which is the subject of the present paper), $C_x(\overline{E})$ is already unique. For general expressions assume that we find a proper representative for each $C_x(\overline{E})$ [9, 16]. Define the equivalence $=_c \subseteq Q_{\rm pos}^2$ by $x_1 =_c x_2$ iff $C_{x_1}(\overline{E}) = C_{x_2}(\overline{E})$. Define the equivalence $\equiv_c \subseteq Q_{\rm pos}^2$ by $x_1 \equiv_c x_2$ iff $\overline{C_{x_1}(\overline{E})} = \overline{C_{x_2}(\overline{E})}$. Each of the equivalence relations is right invariant w.r.t. $M_{\rm pos}(E)$. It is known that

Proposition 5. *(1) $M_{\rm pd}(E) \simeq M_{\rm pos}(E)/_{\equiv_c}$;*
(2) $\overline{M_{\rm pd}(\overline{E})} \simeq M_{\rm pos}(E)/_{=_c}$.

From Propositions 4 and 5 both $M_{\rm pd}(E)$ and $M_{\rm f}(E)$ are always smaller than or equal to $M_{\rm pos}(E)$. However, for the relation between $M_{\rm pd}(E)$ and $M_{\rm f}(E)$, Ilie and Yu [16] compared some examples and showed that it is difficult to give a theoretical analysis. Here we give a theoretical analysis.

First we present some simple results. It is easy to see that

Lemma 1. *For any $a \in \Sigma_{\overline{E}}$,*
(1) $first(C_a(\overline{E})) = follow(\overline{E}, a)$ ([2]), and (2) $a \in last_0(\overline{E}) \Leftrightarrow \lambda(C_a(\overline{E})) = \varepsilon$.

From Lemma 1 and the above definitions of the equivalence relations, the following are implied

Lemma 2. *(1) $=_c \subseteq \equiv_f$; (2) $=_c \subseteq \equiv_c$.*

Then we give a characterization of $=_c = \equiv_f$ as follows.

Proposition 6. *For an expression E, we have $=_c = \equiv_f$ iff $\forall a, b \in Q_{\rm pos}$, $first(C_a(\overline{E})) = first(C_b(\overline{E})) \wedge \lambda(C_a(\overline{E})) = \lambda(C_b(\overline{E})) \Rightarrow C_a(\overline{E}) = C_b(\overline{E})$.*

Proof. $=_c = \equiv_f$ iff $=_c \subseteq \equiv_f \wedge \equiv_f \subseteq =_c$
iff true $\wedge \equiv_f \subseteq =_c$ (Lemma 2 (1))
iff $\equiv_f \subseteq =_c$
iff $\forall a, b \in Q_{\rm pos}$, $follow(\overline{E}, a) = follow(\overline{E}, b) \wedge (a \in last_0(\overline{E}) \Leftrightarrow b \in last_0(\overline{E})) \Rightarrow C_a(\overline{E}) = C_b(\overline{E})$
iff $\forall a, b \in Q_{\rm pos}$, $first(C_a(\overline{E})) = first(C_b(\overline{E})) \wedge (\lambda(C_a(\overline{E})) = \varepsilon \Leftrightarrow (\lambda(C_b(\overline{E})) = \varepsilon) \Rightarrow C_a(\overline{E}) = C_b(\overline{E})$ (Lemma 1)
iff $\forall a, b \in Q_{\rm pos}$, $first(C_a(\overline{E})) = first(C_b(\overline{E})) \wedge \lambda(C_a(\overline{E})) = \lambda(C_b(\overline{E})) \Rightarrow C_a(\overline{E}) = C_b(\overline{E})$. □

Similarly the following is a characterization of $=_c = \equiv_c$.

Proposition 7. *For an expression E, we have $=_c \; = \; \equiv_c$ iff $\forall a, b \in Q_{\rm pos}$, $\overline{C_a(\overline{E})} = \overline{C_b(\overline{E})} \Rightarrow C_a(\overline{E}) = C_b(\overline{E})$.*

The proof is similar to the proof of Proposition 6.

And the following is a characterization of $\equiv_c = \equiv_f$.

Proposition 8. *For an expression E, we have $\equiv_c \; = \; \equiv_f$ iff $\forall a, b \in Q_{\rm pos}$, $\overline{C_a(\overline{E})} = \overline{C_b(\overline{E})} \Leftrightarrow first(C_a(\overline{E})) = first(C_b(\overline{E})) \wedge \lambda(C_a(\overline{E})) = \lambda(C_b(\overline{E}))$.*

The proof is similar to the proof of Proposition 6.

On the other hand, from Propositions 4,5 and Lemma 2 it follows

Proposition 9. *For an expression E,*
(1) if $=_c = \equiv_f$, then $M_{\rm pd}(E)$ is a quotient of $M_{\rm f}(E)$, $\overline{M_{\rm pd}(\overline{E})} \simeq M_{\rm f}(E)$; and
(2) if $=_c = \equiv_c$, then $M_{\rm f}(E)$ is a quotient of $M_{\rm pd}(E)$, $\overline{M_{\rm pd}(\overline{E})} \simeq M_{\rm pd}(E)$; and
(3) if $\equiv_c = \equiv_f$, then $M_{\rm pd}(E) \simeq M_{\rm f}(E)$.

Proof. We prove (1) only, others are proved similarly. Let $M_{\rm pos}(E) = (Q, \Sigma, \delta, q_0, F)$, then $M_{\rm f}(E) \simeq (Q/_{\equiv_f}, \Sigma, \delta_{\equiv_f}, [q_0]_{\equiv_f}, F/_{\equiv_f})$, $M_{\rm pd}(E) \simeq (Q/_{\equiv_c}, \Sigma, \delta_{\equiv_c}, [q_0]_{\equiv_c}, F/_{\equiv_c})$. Define equivalence $\cong \subseteq Q/_{\equiv_f} \times Q/_{\equiv_f}$ by $[a_i]_{\equiv_f} \cong [a_j]_{\equiv_f}$ iff $a_i \equiv_c a_j$ for $a_i, a_j \in Q$. Since $\equiv_f \subseteq \equiv_c$, $\cong$ is well-defined. It is easy to know $\cong$ is right invariant. Then $M_{\rm pd}(E) \simeq M_{\rm f}(E)/_{\cong}$. From Propositions 5,4, $\overline{M_{\rm pd}(\overline{E})} \simeq M_{\rm pos}(E)/_{=_c} = M_{\rm pos}(E)/_{\equiv_f} \simeq M_{\rm f}(E)$. □

Example 3. Let $E_1 = aa^* + ba^*, E_2 = (a^* + \varepsilon)a^*a^*, E_3 = a^*$, one can verify that $M_{\rm pd}(E_1)$ is a quotient of $M_{\rm f}(E_1)$, $M_{\rm f}(E_2)$ is a quotient of $M_{\rm pd}(E_2)$, and $M_{\rm pd}(E_3) \simeq M_{\rm f}(E_3)$.

The above characterizations are given in terms of $C_x(\overline{E})$. Below we consider conditions in terms of the structure of expressions.

We first prove the following Lemmas. Recall that we assume that the rules (rules-$\emptyset\varepsilon$) hold.

It is known that the following property holds:

$first(\overline{F+G}) = first(\overline{F}) \cup first(\overline{G}), first(\overline{F^*}) = first(\overline{F})$,
$first(\overline{FG}) = first(\overline{F}) \cup first(\overline{G})$ if $\varepsilon \in L(F)$, $first(\overline{F})$ otherwise.
$last(\overline{F+G}) = last(\overline{F}) \cup last(\overline{G}), last(\overline{F^*}) = last(\overline{F})$,
$last(\overline{FG}) = last(\overline{F}) \cup last(\overline{G})$ if $\varepsilon \in L(G)$, $last(\overline{G})$ otherwise.

$$follow(\overline{F+G}, a) = \begin{cases} follow(\overline{F}, a), \text{ if } a \in \Sigma_{\overline{F}} \\ follow(\overline{G}, a), \text{ if } a \in \Sigma_{\overline{G}} \end{cases}$$

$$follow(\overline{FG}, a) = \begin{cases} follow(\overline{F}, a), & \text{if } a \in \Sigma_{\overline{F}} - last(\overline{F}) \\ follow(\overline{F}, a) \cup first(\overline{G}), & \text{if } a \in last(\overline{F}) \\ follow(\overline{G}, a), & \text{if } a \in \Sigma_{\overline{G}} \end{cases}$$

$$follow(\overline{F^*}, a) = \begin{cases} follow(\overline{F}, a), & \text{if } a \in \Sigma_{\overline{F}} - last(\overline{F}) \\ follow(\overline{F}, a) \cup first(\overline{F}), & \text{if } a \in last(\overline{F}) \end{cases}$$

Lemma 3. *For $b \in \Sigma_{\overline{E}}$, $follow(\overline{E}, b) = \emptyset$ iff $\forall w \in \Sigma^*_{\overline{E}}$, $(wb)^{-1}(\overline{E}) = \emptyset$ or $(wb)^{-1}(\overline{E}) = \varepsilon$.*

Proof. (Only if). Since $follow(\overline{E}, b) = \emptyset$, it must have $b \in last(\overline{E})$, and b can only appear in the last position of a word of $L(\overline{E})$. Then $\forall w \in \Sigma^*_{\overline{E}}$, if $wb \notin L(\overline{E})$, then $(wb)^{-1}(\overline{E}) = \emptyset$. Otherwise, $(wb)^{-1}(\overline{E}) = \varepsilon$.
(If). If $\forall w \in \Sigma^*_{\overline{E}}$, $(wb)^{-1}(\overline{E}) = \emptyset$ or $(wb)^{-1}(\overline{E}) = \varepsilon$, then $follow(\overline{E}, b) = first((wb)^{-1}(\overline{E})) = \emptyset$. □

For an expression E, the leftmost expression of E w.r.t. concatenation is $le(E) = le(F)$ if $E = FG$; E otherwise. We say an expression E is leftmost ε-reduced if $le(E)$ does not contain any subexpression $F + \varepsilon$ where $\lambda(F) = \varepsilon$. Obviously if $E = FG$ then E is leftmost ε-reduced iff F is leftmost ε-reduced.

Lemma 4. *If an expression E is leftmost ε-reduced and in SNF, $b \in \Sigma_{\overline{E}}$ and E satisfies the following condition:*

$$\begin{array}{l}(1)\ b \in \Sigma_{\overline{E}} - last(\overline{E}) \text{ and } \varepsilon \notin L(\overline{E}),\ \text{or } b \in last(\overline{E}) \text{ and } \varepsilon \in L(\overline{E}),\ \text{and} \\ (2)\ first(\overline{E}) = follow(\overline{E}, b),\end{array} \tag{3}$$

then E can be only of the form F^* or $T_n^* G_n \dots G_0, n \geq 0$, where $b \in \Sigma_{\overline{T_n}}$, and T_n^* satisfies Condition (3); and $\forall w \in \Sigma_{\overline{E}}^*$, if $(wb)^{-1}(\overline{E}) \neq \emptyset$ then $(wb)^{-1}(\overline{E}) = \overline{E}$.

Proof. If E satisfies Condition (3), it is obvious that $\overline{E}$ cannot be $\emptyset, \varepsilon$ or $a \in \Sigma_{\overline{E}}$.

We show $\overline{E}$ cannot be $\overline{F} + \overline{G}$.

Let $\overline{E} = \overline{F} + \overline{G}$, then $first(\overline{E}) = first(\overline{F}) \cup first(\overline{G})$. Assume $b \in \Sigma_{\overline{F}}$, then $follow(\overline{E}, b) = follow(\overline{F}, b)$. So by Condition (3), $first(\overline{F}) \cup first(\overline{G}) = follow(\overline{F}, b)$, then $first(\overline{F}) = follow(\overline{F}, b)$ and $first(\overline{G}) = \emptyset$. Since $first(\overline{G}) = \emptyset$, $\overline{G} = \emptyset$ or ε. Since $\overline{E}$ is reduced by rules (rules-$\emptyset\varepsilon$), $\overline{G}$ cannot be $\emptyset$. So $\overline{G} = \varepsilon$, then $\overline{E} = \overline{F} + \varepsilon, \varepsilon \in L(\overline{E})$, and since $\overline{E}$ is leftmost ε-reduced, $\varepsilon \notin L(\overline{F})$. From $\varepsilon \in L(\overline{E})$, by Condition (3), we have $b \in last(\overline{E})$, then $b \in last(\overline{F})$. We show this is impossible. Since $\varepsilon \notin L(\overline{F})$ and $b \in last(\overline{F})$, $\overline{F}$ cannot be $\emptyset, \varepsilon$ or $\overline{I}^*$. If $\overline{F} = b$, $first(\overline{F}) \neq follow(\overline{F}, b)$, which is a contradiction. If $\overline{F} = \overline{I} + \overline{J}$, suppose $b \in \Sigma_{\overline{I}}$, similarly as above,we can deduce $\overline{J} = \varepsilon$. But this contradicts with $\varepsilon \notin L(\overline{F})$. If $\overline{F} = \overline{IJ}$, then if $b \in \Sigma_{\overline{I}}$, since $b \in last(\overline{F}), \varepsilon \in L(\overline{J})$. Since $\varepsilon \notin L(\overline{F})$, $\varepsilon \notin L(\overline{I})$. Then $follow(\overline{F}, b) = follow(\overline{I}, b) \cup first(\overline{J}), first(\overline{F}) = first(\overline{I})$. So $first(\overline{J}) = \emptyset, \overline{J} = \emptyset$ or ε, but this contradicts with $\overline{F}$ being reduced by rules (rules-$\emptyset\varepsilon$). If $b \in \Sigma_{\overline{J}}$, then $first(\overline{F}) = first(\overline{I}) \cup \lambda(\overline{I}) first(\overline{J}), follow(\overline{F}, b) = follow(\overline{J}, b)$. So $first(\overline{I}) = \emptyset, \overline{I} = \emptyset$ or ε, but this contradicts with $\overline{F}$ being reduced by rules (rules-$\emptyset\varepsilon$). If $b \in \Sigma_{\overline{G}}$, the proof is similar.

If $\overline{E} = \overline{F^*}$, since $\varepsilon \in L(\overline{E})$, by Condition (3), $b \in last(\overline{E})$, then $b \in last(\overline{F})$. So $follow(\overline{E}, b) = follow(\overline{F}, b) \cup first(\overline{F}), first(\overline{E}) = first(\overline{F})$. Since $\overline{E}$ is in star normal form, we have $follow(\overline{F}, b) \cap first(\overline{F}) = \emptyset$ for $b \in last(\overline{F})$. So $follow(\overline{F}, b) = \emptyset$. Then from Lemma 3, $\forall w \in \Sigma_{\overline{F}}^*$, if $(wb)^{-1}(\overline{F}) \neq \emptyset$ then $(wb)^{-1}(\overline{F}) = \varepsilon$, so $(wb)^{-1}(\overline{E}) = (wb)^{-1}(\overline{F})\overline{F^*} = \overline{E}$.

If $\overline{E} = \overline{FG}$, then $first(\overline{E}) = first(\overline{F}) \cup \lambda(\overline{F}) first(\overline{G})$. If $b \in \Sigma_{\overline{F}} - last(\overline{F})$, then $follow(\overline{E}, b) = follow(\overline{F}, b)$. So if $\varepsilon \notin L(\overline{F})$, then $first(\overline{F}) = follow(\overline{F}, b)$. Otherwise $\varepsilon \in L(\overline{F})$, then $first(\overline{G}) = \emptyset$, $\overline{G} = \emptyset$ or ε. But this contradicts with $\overline{E}$ being reduced by rules (rules-$\emptyset\varepsilon$). If $b \in last(\overline{F})$, then $follow(\overline{E}, b) = follow(\overline{F}, b) \cup first(\overline{G})$. Then if $\varepsilon \notin L(\overline{F})$, then $first(\overline{G}) = \emptyset$. Similarly this contradicts with $\overline{E}$ being reduced by rules (rules-$\emptyset\varepsilon$). Otherwise $\varepsilon \in L(\overline{F})$, then $first(\overline{F}) = follow(\overline{F}, b)$. If $b \in \Sigma_{\overline{G}}$, then $follow(\overline{E}, b) = follow(\overline{G}, b)$. So $first(\overline{F}) = \emptyset, \overline{F} = \emptyset$ or ε. Similarly this contradicts with $\overline{E}$ being reduced by rules (rules-$\emptyset\varepsilon$).

So we have F is leftmost ε-reduced and satisfies Condition (3), $b \in \Sigma_{\overline{F}}$. Then repeat the above reasoning to $\overline{F}$, $\overline{F}$ must be of the form $\overline{T_0^*}$ or $\overline{F_1 G_1}$, and repeatedly proceed if necessary .... Since $|\overline{E}|$ is finite, the iteration must terminate, and $\overline{E}$ must be $\overline{E} = \overline{T_n^* G_n} \dots \overline{G_0}, n \geq 0$, where $G_0 = G$, $b \in \Sigma_{\overline{T_n}}$, and T_n^* satisfies Condition (3). Then, $(wb)^{-1}(\overline{E}) = (wb)^{-1}(\overline{T_n^*})\overline{G_n} \dots \overline{G_0} = \overline{E}$. □

Conversely we have

Lemma 5. *If $\forall b \in \Sigma_{\overline{E}}, \forall w \in \Sigma_{\overline{E}}^*$, whenever $(wb)^{-1}(\overline{E}) \neq \emptyset$ then $(wb)^{-1}(\overline{E}) = \overline{E}$, then E satisfies Condition (3).*

Proof. If $\forall b \in \Sigma_{\overline{E}}, \forall w \in \Sigma_{\overline{E}^*}$, whenever $(wb)^{-1}(\overline{E}) \neq \emptyset$ then $(wb)^{-1}(\overline{E}) = \overline{E}$, then for any non-null $(wb)^{-1}(\overline{E}), first(\overline{E}) = first((wb)^{-1}(\overline{E})) = follow(\overline{E}, b)$. Furthermore, if $b \in \Sigma_{\overline{E}} - last(\overline{E})$, then $wb \notin L(\overline{E})$, thus $\varepsilon \notin L((wb)^{-1}(\overline{E})) = L(\overline{E})$. If $b \in last(\overline{E})$, then $\varepsilon \in L((wb)^{-1}(\overline{E})) = L(\overline{E})$. □

For an expression E, we call the following the *CSE (Concatenation Sub-Expressions)* condition of E:

If E satisfies Condition (3) then E is leftmost ε-reduced; For any subexpression FG of E, if $follow(\overline{F}, a) = \emptyset$ and G satisfies Condition (3) for some $a \in \Sigma_{\overline{F}}$ and $b \in \Sigma_{\overline{G}}$ then G is leftmost ε-reduced.

The significance of *CSE* condition and Condition (3) can be seen from the following two lemmas.

Lemma 6. *For $E = FG$, $a \in \Sigma_{\overline{F}}$ and $b \in \Sigma_{\overline{G}}$, $a \equiv_f b$ iff $follow(\overline{F}, a) = \emptyset$ and G satisfies Condition (3).*

Proof. (Only if). If $a \equiv_f b$, it is not difficult to prove that $a \in last(\overline{F})$, and (1) $b \in \Sigma_{\overline{G}} - last(\overline{G})$ and $\varepsilon \notin L(G)$ or (2) $b \in last(\overline{G})$ and $\varepsilon \in L(G)$, then $follow(\overline{E}, a) = follow(\overline{F}, a) \cup first(\overline{G})$, $follow(\overline{E}, b) = follow(\overline{G}, b)$. So $follow(\overline{F}, a) = \emptyset$, and $first(\overline{G}) = follow(\overline{G}, b)$. Then G satisfies Condition (3).

(If). Since $follow(\overline{F}, a) = \emptyset$, $a \in last(\overline{F})$. So $follow(\overline{E}, a) = follow(\overline{F}, a) \cup first(\overline{G}) = first(\overline{G})$. Since G satisfies Condition (3), $follow(\overline{E}, b) = follow(\overline{G}, b) = follow(\overline{E}, a)$. Furthermore, by Condition (3), if $b \in \Sigma_{\overline{G}} - last(\overline{G})$ and $\varepsilon \notin L(G)$, then $a, b \notin last(\overline{E})$; if $b \in last(\overline{G})$ and $\varepsilon \in L(G)$, then $a, b \in last(\overline{E})$.

So $a \equiv_f b$. □

Lemma 7. *For a SNF expression $E = FG$, $a \in \Sigma_{\overline{F}}$ and $b \in \Sigma_{\overline{G}}$,*
if $a =_c b$ then $follow(\overline{F}, a) = \emptyset$ and G satisfies Condition (3); and
if $follow(\overline{F}, a) = \emptyset$, G satisfies Condition (3) and is leftmost ε-reduced then $a =_c b$.

Proof. By Eq (2), for any non-null $(w_1 a)^{-1}(\overline{E}), (w_2 b)^{-1}(\overline{E})$,

$(w_1 a)^{-1}(\overline{E}) = (w_1 a)^{-1}(\overline{F})\overline{G}, (w_2 b)^{-1}(\overline{E}) = (vb)^{-1}(\overline{G})$ for some $w_2 = uv$.

If $a =_c b$, then $(w_1 a)^{-1}(\overline{F}) = \varepsilon$ and $(vb)^{-1}(\overline{G}) = \overline{G}$. By $(w_1 a)^{-1}(\overline{F}) = \varepsilon$ and Lemma 3, $follow(\overline{F}, a) = \emptyset$. By $(vb)^{-1}(\overline{G}) = \overline{G}$ and Lemma 5, G satisfies Condition (3).

If $follow(\overline{F}, a) = \emptyset$, G satisfies Condition (3) and is leftmost ε-reduced, then from $follow(\overline{F}, a) = \emptyset$ and Lemma 3, $\forall w \in \Sigma^*_{\overline{F}}$, if $(wa)^{-1}(\overline{F}) \neq \emptyset$ then $(wa)^{-1}(\overline{F}) = \varepsilon$. Since G satisfies Condition (3) and is leftmost ε-reduced, and G is in SNF, from Lemma 4, $\forall w \in \Sigma^*_{\overline{G}}$, if $(wb)^{-1}(\overline{G}) \neq \emptyset$ then $(wb)^{-1}(\overline{G}) = \overline{G}$. So $(w_1 a)^{-1}(\overline{E}) = (w_1 a)^{-1}(\overline{F})\overline{G} = \overline{G}$, $(w_2 b)^{-1}(\overline{E}) = (vb)^{-1}(\overline{G}) = \overline{G}$. Then $a =_c b$. □

The following is a sufficient condition for $=_c = \equiv_f$.

Proposition 10. *For a SNF expression E satisfying CSE condition, we have $=_c = \equiv_f$.*

Proof. By Lemma 2(1), we only need to prove that $\equiv_f \subseteq =_c$.

Let $a \equiv_f b$.

(1) If $a = q_E$, we show that $a =_c b$. If $b = q_E$ then $a =_c b$. Otherwise, if $a \in last_0(\overline{E})$, then $b \in last(\overline{E})$ and $\varepsilon \in L(E)$, otherwise $b \in \Sigma_{\overline{E}} - last(\overline{E})$ and $\varepsilon \notin L(E)$. Furthermore, $follow(\overline{E}, b) = first(\overline{E})$. Then E satisfies Condition (3), and so is leftmost ε-reduced. From Lemma 4, $(wb)^{-1}(\overline{E}) = \overline{E}$ if $(wb)^{-1}(\overline{E}) \neq \emptyset$. Therefore $C_b(\overline{E}) = \overline{E} = C_a(\overline{E})$.

(2) Similarly if $b = q_E$, we have $a =_c b$.

(3) Below we consider situations when $a \neq q_E$ and $b \neq q_E$.

Now we prove if $a \equiv_f b$ then $a =_c b$ by induction on the structure of $\overline{E}$.

Base. If $\overline{E} = \emptyset, \varepsilon$, or a, this is obvious.

Induction. 1. $\overline{E} = \overline{F + G}$. Without losing generality, suppose $a \in \Sigma_{\overline{F}}$, then $follow(\overline{F + G}, a) = follow(\overline{F}, a)$. If $b \in \Sigma_{\overline{F}}$, then $follow(\overline{F + G}, b) = follow(\overline{F}, b)$. By the induction hypothesis, for any non-null $(w_1 a)^{-1}(\overline{F})$ and $(w_2 b)^{-1}(\overline{F})$, $(w_1 a)^{-1}(\overline{F}) = (w_2 b)^{-1}(\overline{F})$. Note any non-null $(w_1 a)^{-1}(\overline{F})$ is unique and is $C_a(\overline{F})$ since F is in SNF. Since $(w_1 a)^{-1}(\overline{E}) = (w_1 a)^{-1}(\overline{F})$ and $(w_2 b)^{-1}(\overline{E}) = (w_2 b)^{-1}(\overline{F})$, $a =_c b$. Otherwise b is in $\Sigma_{\overline{G}}$. So $follow(\overline{F + G}, b) = follow(\overline{G}, b)$. It is only possible that $follow(\overline{F + G}, a) = follow(\overline{F + G}, b) = \emptyset$. Since $follow(\overline{F}, a) = \emptyset$, from Lemma 3, $\forall w \in \Sigma^*_{\overline{F}}$, if $(wa)^{-1}(\overline{F}) \neq \emptyset$ then $(wa)^{-1}(\overline{F}) = \varepsilon$. So, $(wa)^{-1}(\overline{E}) = (wa)^{-1}(\overline{F}) = \varepsilon$. Similarly, $\forall w \in \Sigma^*_{\overline{G}}$, if $(wb)^{-1}(\overline{E}) \neq \emptyset$, then $(wb)^{-1}(\overline{E}) = \varepsilon$. So $a =_c b$.

2. $\overline{E} = \overline{FG}$. It is not difficult to prove that (i) if $a \in \Sigma_{\overline{F}} - last(\overline{F})$, then $b \in \Sigma_{\overline{F}} - last(\overline{F})$, (ii) if $a \in last(\overline{F})$, then (a) $b \in last(\overline{F})$, (b) $b \in \Sigma_{\overline{G}} - last(\overline{G})$ and $\varepsilon \notin L(G)$ or (c) $b \in last(\overline{G})$ and

$\varepsilon \in L(G)$, (iii) if $a \in \Sigma_{\overline{G}} - last(\overline{G})$, then (a) $b \in last(\overline{F})$ and $\varepsilon \notin L(G)$ or (b) $b \in \Sigma_{\overline{G}} - last(\overline{G})$, and (iv) if $a \in last(\overline{G})$, then (a) $b \in last(\overline{F})$ and $\varepsilon \in L(G)$ or (b) $b \in last(\overline{G})$. For example, if $a \in \Sigma_{\overline{F}} - last(\overline{F})$, then $a \notin last(\overline{FG})$. So (a) $b \in \Sigma_{\overline{F}} - last(\overline{F})$, (b) $b \in last(\overline{F})$ and $\varepsilon \notin L(G)$ or (c) $b \in \Sigma_{\overline{G}} - last(\overline{G})$. Suppose $G \neq \emptyset, \varepsilon$, otherwise E is reduced to $\emptyset$ or F by rules (rules-$\emptyset\varepsilon$). If (b), $follow(\overline{FG}, a) = follow(\overline{F}, a)$, $follow(\overline{FG}, b) = follow(\overline{F}, b) \cup first(\overline{G})$. Since $follow(\overline{F}, a) \subseteq \Sigma_{\overline{F}}$, $first(\overline{G}) = \emptyset$, but this is impossible if $G \neq \emptyset, \varepsilon$. If (c), $follow(\overline{F}, a) = follow(\overline{G}, b)$, which is possible only if $follow(\overline{F}, a) = \emptyset$, but this is impossible for $a \in \Sigma_{\overline{F}} - last(\overline{F})$. Other proofs are left to readers.

If (i), then $follow(\overline{FG}, a) = follow(\overline{F}, a)$, $follow(\overline{FG}, b) = follow(\overline{F}, b)$. By the induction hypothesis, similarly to 1, we can prove $a =_c b$. If (ii.a), then $follow(\overline{FG}, a) = follow(\overline{F}, a) \cup first(\overline{G})$, $follow(\overline{FG}, b) = follow(\overline{F}, b) \cup first(\overline{G})$. Since $follow(\overline{F}, a) \cap first(\overline{G}) = \emptyset$, and $follow(\overline{F}, b) \cap first(\overline{G}) = \emptyset$, we have $follow(\overline{F}, a) = follow(\overline{F}, b)$. By the induction hypothesis, similarly we can prove $a =_c b$. If (ii.b) or (ii.c), then from Lemma 6, $follow(\overline{F}, a) = \emptyset$ and G satisfies Condition (3). By *CSE* condition G is leftmost ε-reduced, then from Lemma 7 we have $a =_c b$.

The proofs for (iii) and (iv) can be given similarly (In fact, (iii.a) and (iv.a) are symmetrical to (ii.b) and (ii.c), respectively.).

3. $\overline{E} = \overline{F^*}$. If $a, b \in \Sigma_{\overline{F}} - last(\overline{F})$, then $follow(\overline{F^*}, a) = follow(\overline{F}, a)$, and $follow(\overline{F^*}, b) = follow(\overline{F}, b)$. By the induction hypothesis, similarly as before we can prove $a =_c b$. If $a, b \in last(\overline{F})$, then $follow(\overline{F^*}, a) = follow(\overline{F}, a) \cup first(\overline{F})$, $follow(\overline{F^*}, b) = follow(\overline{F}, b) \cup first(\overline{F})$. Since E is in SNF, $followlast(\overline{F}) \cap first(\overline{F}) = \emptyset$, so $follow(\overline{F}, a) \cap first(\overline{F}) = \emptyset$, $follow(\overline{F}, b) \cap first(\overline{F}) = \emptyset$ for $a, b \in last(\overline{F})$. Then $follow(\overline{F}, a) = follow(\overline{F}, b)$. By the induction hypothesis, similarly we can prove $a =_c b$. □

Note the restriction that final and non-final states cannot be $\equiv_f$-equivalent is essential, as shown by the expression $E = b^*a(b^*a)^*$. Let $\overline{E} = b_1^* a_2 (b_3^* a_4)^*$. Then $C_{a_2}(\overline{E}) = C_{a_4}(\overline{E}) \neq C_{b_3}(\overline{E})$, $follow(\overline{E}, a_2) = follow(\overline{E}, b_3) = follow(\overline{E}, a_4)$. However, $a_2, a_4 \in last(\overline{E})$ and $b_3 \notin last(\overline{E})$.

Corollary 3. *For a SNF regular expression E satisfying CSE condition, $\overline{M_{\rm pd}(\overline{E})} \simeq M_{\rm f}(E)$.*

Corollary 4. *For a SNF regular expression E satisfying CSE condition, $M_{\rm pd}(E)$ is a quotient of $M_{\rm f}(E)$.*

Example 4. Let $E_1 = a^*(a^* + \varepsilon)c^*, E_2 = a(a^* + \varepsilon + b), E_3 = (a^* + \varepsilon)b^*c^* + d$, they all are in SNF and satisfy *CSE* condition. One can verify that for each E_i, $=_c = \equiv_f$, and $M_{\rm pd}(E_i)$ is a quotient of $M_{\rm f}(E_i), i = 1, 2, 3$.

Also, the expressions E_1 in Example 1, E_1 and E_3 in Example 3 are in SNF and satisfy *CSE* condition, and their equation automata are quotients of the follow automata.

We further present the following conditions for some special situations, concerning also $=_c$ and $\equiv_c$.

Condition 1. Let $E = F_1 F_2 \dots F_n, F_r$ is of the form: $a, a^*, a^* + \varepsilon$ or $\varepsilon + a^*, a \in \Sigma, r = 1, \dots, n, n \geq 1$.
(a) F_1 is of the form a or a^*, and
(b) if $F_r = a$, then F_{r+1} is of the form b or b^*.

Proposition 11. *For a regular expression E satisfying Condition 1, we have $=_c = \equiv_f$, $=_c = \equiv_c$, and $\equiv_c = \equiv_f$.*

Proof. Since E is in SNF and satisfies *CSE* condition, by Proposition 10, $=_c = \equiv_f$.

Now we prove $=_c = \equiv_c$. $\overline{E} = \overline{F_1 F_2} \dots \overline{F_n}$, $\forall a, b \in Q_{\rm pos}, a \neq b$, write $a < b$ if $a = q_E, b \in \overline{E}$ or $a \in \overline{F_i}, b \in \overline{F_j}, i < j$. Without losing generality suppose $a < b$. Suppose $\overline{C_a(\overline{E})} = \overline{C_b(\overline{E})}$. This is possible only if $a = q_E$ and $b \in \overline{F_1}$, or $a \in \overline{F_r}$ and $b \in \overline{F_{r+1}}$, otherwise $|\overline{C_a(\overline{E})}| \neq |\overline{C_b(\overline{E})}|$, hence $\overline{C_a(\overline{E})} \neq \overline{C_b(\overline{E})}$. Furthermore, if $a = q_E$, then $\overline{F_1}$ is b^*, and $C_a(\overline{E}) = C_b(\overline{E}) = \overline{E}$. If $a \in \overline{F_r}$, then $\overline{F_r} = a$, and $\overline{F_{r+1}}$ is b^*, and $C_a(\overline{E}) = C_b(\overline{E}) = \overline{F_{r+1}} \dots \overline{F_n}$. So from Proposition 7, $=_c = \equiv_c$.

Then $\equiv_c = \equiv_f$ follows. □

Condition 2. Let $E = F_1F_2 \dots F_n, n \geq 1$ the same as in Condition 1. The following is satisfied at least once:
(a) F_1 is of the form $a^* + \varepsilon$ or $\varepsilon + a^*$, or
(b) if $F_r = a$, then F_{r+1} is of the form $b^* + \varepsilon$ or $\varepsilon + b^*$.

Note Condition 2 is the negated one of Condition 1 w.r.t. E.

Proposition 12. *For a regular expression E satisfying Condition 2, we have $=_c \neq \equiv_f$ and $=_c = \equiv_c$.*

The proof of Proposition 12 is similar to the proof of Proposition 11.

Corollary 5. *For a regular expression E satisfying Condition 1, $M_{\mathrm{pd}}(E) \simeq M_{\mathrm{f}}(E) \simeq \overline{M_{\mathrm{pd}}(\overline{E})}$. For a regular expression E satisfying Condition 2, $M_{\mathrm{pd}}(E) \not\simeq M_{\mathrm{f}}(E), \overline{M_{\mathrm{pd}}(\overline{E})} \simeq M_{\mathrm{pd}}(E)$, and $M_{\mathrm{f}}(E)$ is a quotient of $M_{\mathrm{pd}}(E)$.*

For example, the expressions E_3 in Example 3 and E_1 in Example 4 satisfy Condition 1, and their equation and follow automata are isomorphic. The expression E_2 in Example 3 satisfies Condition 2, and its follow automaton is a quotient of the equation automaton.

If an expression E is linear, there is a one-one correspondence between the symbols in E and $\overline{E}$. Then for $C_a(\overline{E}) \neq C_b(\overline{E})$ it cannot be $\overline{C_a(\overline{E})} = \overline{C_b(\overline{E})}$. So

Proposition 13. *For a linear expression E, we have $=_c = \equiv_c$.*

Corollary 6. *For a linear expression E, $\overline{M_{\mathrm{pd}}(\overline{E})} \simeq M_{\mathrm{pd}}(E)$, and $M_{\mathrm{f}}(E)$ is a quotient of $M_{\mathrm{pd}}(E)$.*

For example, the expression E_1 in Example 1 is linear, so $=_c = \equiv_c$. We also know that for E_1 we have $=_c = \equiv_f$. Therefore $\equiv_c = \equiv_f$, that is, $M_{\mathrm{pd}}(E_1) \simeq M_{\mathrm{f}}(E_1)$. Similarly, for the expression E_3 in Example 4, we have $M_{\mathrm{pd}}(E_3) \simeq M_{\mathrm{f}}(E_3)$ since it is linear and as we know from Example 4 for E_3 we have $=_c = \equiv_f$.

So far we have presented some conditions for the relations among $=_c, \equiv_c$ and $\equiv_f$, hence the relations between $M_{\mathrm{pd}}(E)$ and $M_{\mathrm{f}}(E)$. Since regular expressions can be transformed to SNF in linear time [4], we can easily get the smaller automaton when one of the above conditions is satisfied. It would be interesting to find some more conditions, which remains as a further research.

5 One-unambiguous expressions

5.1 Properties of automata

As shown by Brüggemann-Klein [4] and Brüggemann-Klein and Wood [6], we have the following properties.

Proposition 14. *(1) E is one-unambiguous iff $M_{\mathrm{pos}}(E)$ is deterministic.*
(2) It can be decided in linear time whether E is one-unambiguous.

For a one-unambiguous expression E, from Proposition 14(1) and Propositions 4 and 5 it is easy to have

Proposition 15. *If a regular expression E is one-unambiguous, then $M_{\mathrm{pd}}(E)$ ($M_{\mathrm{f}}(E)$) is deterministic, but not vice versa.*

Proof. For a one-unambiguous expression E, since $M_{\mathrm{pos}}(E)$ is deterministic, and both $M_{\mathrm{pd}}(E)$ and $M_{\mathrm{f}}(E)$ are quotients of $M_{\mathrm{pos}}(E)$, it is implied that $M_{\mathrm{pd}}(E)$ and $M_{\mathrm{f}}(E)$ are deterministic.

Conversely, for a regular expression E, if $M_{\mathrm{pd}}(E)$ is deterministic, E may not be one-unambiguous. This can be shown by the example $E = (c + b)d + cd$, of which $M_{\mathrm{pd}}(E)$ is deterministic, but E is not one-unambiguous. Similarly, if $M_{\mathrm{f}}(E)$ is deterministic, E may not be one-unambiguous. This can be shown by the example $E = a + b^*a$. One can easily verify that $M_{\mathrm{f}}(E)$ is deterministic. However, E is not one-unambiguous. □

Corollary 7. *For a one-unambiguous expression E, we have*
(1) for any word u, $\partial_u(E)$ which is not empty is a singleton set,
(2) $DD(E) \sim PD(E)$,
(3) $|DD(E)| \leq \|E\| + 1$.

Proof. (1) For a one-unambiguous expression E, since $M_{\mathrm{pd}}(E)$ is deterministic by Proposition 15, then the result follows.

(2) By (1), there is a 1-1 function between $DD(E)$ and $PD(E)$.

(3) $|DD(E)| \leq \|E\| + 1$ follows from (2) and Proposition 1. □

Proposition 16. *For a one-unambiguous expression E, we have $M_{\mathrm{dd}}(E) \simeq M_{\mathrm{pd}}(E)$.*

Proof. $M_{\mathrm{dd}}(E) \simeq M_{\mathrm{pd}}(E)$ follows from Corollary 7 (2). □

5.2 Constructing equation and follow automata

For one-unambiguous expressions, it is known that $M_{\mathrm{pos}}(E)$ can be constructed in linear time [4]. Could the efficiency of construction of $M_{\mathrm{pd}}(E)$ or $M_{\mathrm{f}}(E)$ be improved? In the following we answer the question and show that they can also be computed in linear time for one-unambiguous expressions.

Constructing equation automata So far there have been two quadratic time algorithms for construction of the equation automaton. Champarnaud and Ziadi's algorithm [9] is based on the concepts of c-derivatives and c-continuation defined by them, and makes use of the result that the equation automaton is a quotient of the c-continuation automaton. The algorithm consists of computations of the states and the transitions of the quotient of the c-continuation automaton. The key technique is in the sorting of projections of c-continuations used for identifying identical sub-expressions. Khorsi, Ouardi and Ziadi's algorithm [18] is an improvement of the previous one. They avoid the sorting step and replace it by a minimization of an acyclic finite deterministic automaton which requires linear time. Other computations are similar to the previous one.

Proposition 17. *For a one-unambiguous expression E, $M_{\mathrm{pd}}(E)$ can be computed in linear time.*

Proof. Since the equation automaton is a quotient of the position automaton, we sketch the following construction for a regular expression E.

(1) Computation of $M_{\mathrm{pos}}(E)$; (2) Computation of the equivalence relation $\equiv_c$; (3) Computation of $M_{\mathrm{pos}}(E)/_{\equiv_c}$.

(1). It is known that $M_{\mathrm{pos}}(E)$ can be computed in quadratic time, and in linear time if E is one-unambiguous.

(2). We can use the technique for computing equivalence relation on c-derivatives introduced in [18] to compute $\equiv_c$, which is in linear time.

(3). $M_{\mathrm{pos}}(E)/_{\equiv_c}$ can be computed by using standard techniques such as coding and arrays, as follows. In step (1), we can use $1, \ldots, \|E\|$ as subscripts of symbols in $\overline{E}$. Also associate 0 as the subscript for q_E. Denote $pos(x)$ the subscript of x for $x \in \Sigma_{\overline{E}} \cup \{q_E\}$. Then we can use an array $Tr[0..\|E\|, \Sigma_E]$ as the transition table, such that if $(x, a, y) \in \delta_E$ then $Tr[pos(x), a] \leftarrow pos(y)$. In step (2), we associate each equivalence class C of states of $M_{\mathrm{pos}}(E)$ w.r.t. $\equiv_c$ with a unique number n_C, such that all the numbers are consecutive and starting from 0, and keep the coding in another table $Cd[0..\|E\|]$ such that if $x \in C$ then $Cd[pos(x)] = n_C$. Suppose the maximum of the coding is N. The set of states of $M_{\mathrm{pos}}(E)/_{\equiv_c}$ consists of the numbers representing equivalence classes. To compute the transitions of $M_{\mathrm{pos}}(E)/_{\equiv_c}$, we start from the equivalence classes and create a new transition table $Tr/_{\equiv_c}[0..N, \Sigma_E]$ according to Tr and Cd, as follows: First, associate each $Tr/_{\equiv_c}[k, a]$ with an auxiliary array $A_{(k,a)}[0..N]$, of which each element is set to 0 initially. For each equivalence class C, for each $x \in C$, for each r in $Tr[pos(x), a]$, let $t = C_d[r]$, if $A_{(n_C,a)}[t] \neq 1$, then $Tr/_{\equiv_c}[n_C, a] \leftarrow t$, and $A_{(n_C,a)}[t] = 1$. It is easy to see that transitions can be computed in $O(\|E\|^2 \cdot |\Sigma_E|)$ time. If $M_{\mathrm{pos}}(E)$ is deterministic, then the time is $O(\|E\| \cdot |\Sigma_E|)$.

So the above computation is in quadratic time, and in linear time if E is one-unambiguous. □

Constructing follow automata Ilie and Yu presented a quadratic time algorithm [16] to compute follow automaton by constructing an ε-automaton and eliminating ε-transitions from the automaton. Champarnaud, Nicart and Ziadi presented another quadratic construction [8] by computing $\equiv_f$ and the transitions of the follow automaton.

Proposition 18. *For a one-unambiguous expression E, $M_{\mathrm{f}}(E)$ can be computed in linear time.*

Proof. For a one-unambiguous expression E, the follow automaton can be computed very simply. The construction is based on $M_{\mathrm{f}}(E) \simeq M_{\mathrm{pos}}(E)/_{\equiv_f}$. $M_{\mathrm{pos}}(E)$ can be computed in linear time. Since $M_{\mathrm{pos}}(E)$ is deterministic, $follow(\overline{E}, x)$ is either $\emptyset$ or singleton for $x \in \Sigma_{\overline{E}}$. We can use an array $F[0..\|E\|]$ to compute $\equiv_f$. If $follow(\overline{E}, a_r) = a_j$, then $F[pos(a_j)] \leftarrow pos(a_r)$. This can be computed in linear time. Then, similar as in constructing equation automata introduced above, $M_{\mathrm{pos}}(E)/_{\equiv_f}$ can be computed in linear time. So the computation is in linear time. □

6 One-unambiguous expressions in SNF

When expressions are both one-unambiguous and in SNF, we also have the following constructions and properties for derivatives and partial derivatives. Note we assume that the rules (rules-$\emptyset\varepsilon$) hold in the paper.

Proposition 19. *(1) For a one-unambiguous expression E and a symbol a, $a^{-1}(E)$ can be computed as follows:*

$$
\begin{aligned}
a^{-1}(\emptyset) &= a^{-1}(\varepsilon) = \emptyset \\
a^{-1}(b) &= \begin{cases} \varepsilon, \text{ if } b = a \\ \emptyset, \text{ otherwise} \end{cases} \\
a^{-1}(F+G) &= \begin{cases} a^{-1}(F), \text{ if } a \in first(F) \\ a^{-1}(G), \text{ if } a \in first(G) \\ \emptyset, \quad \text{otherwise} \end{cases} \\
a^{-1}(FG) &= \begin{cases} a^{-1}(F)G, \text{ if } a \in first(F) \\ a^{-1}(G), \quad \text{if } a \in first(G) \text{ and } \varepsilon \in L(F) \\ \emptyset, \quad \text{otherwise} \end{cases} \\
a^{-1}(F^*) &= a^{-1}(F)F^*
\end{aligned}
$$

(2) For a one-unambiguous expression E in SNF, the derivatives of E with respect to words can be computed by application of the equations in (1).

Proof. (1) We need to prove only for $E = F + G$ or FG.

If $E = F + G$, then $first(F) \cap first(G) = \emptyset$ by [6]. So if $a \in first(F)$ then $a \notin first(G)$, then $a^{-1}(G) = \emptyset$, $a^{-1}(E) = a^{-1}(F)$. The same is for $a \in first(G)$. If both $a \notin first(F)$ and $a \notin first(G)$, then $a^{-1}(E) = \emptyset$.

If $E = FG$, by definition $a^{-1}(E) = a^{-1}(F)G + a^{-1}(G)$ if $\varepsilon \in L(F)$, or $a^{-1}(F)G$ otherwise. Consider $\varepsilon \in L(F)$. First we show $first(F) \cap first(G) = \emptyset$. If $L(E) = \emptyset$, obviously $first(F) \cap first(G) = \emptyset$, otherwise $L(F), L(G) \neq \emptyset$ which means $L(E) \neq \emptyset$. If $L(E) \neq \emptyset$, then from [6] $first(F) \cap first(G) = \emptyset$. Then, the remaining proof is similar to that of the above for $E = F + G$. If $\varepsilon \notin L(F)$, then if $a \in first(F)$, $a^{-1}(E) = a^{-1}(F)G$, otherwise $a^{-1}(E) = \emptyset$.

(2) If a one-unambiguous expression E is in SNF, it is known [6] that $a^{-1}(E)$ is one-unambiguous and in SNF, then the computation of derivatives of E with respect to words can always use the equations in (1). □

Note the above is an improved result in [12]. In [12] to define $a^{-1}(E)$ it is required that E is in SNF. This is actually unnecessary. SNF is only necessary for derivatives of E with respect to words to ensure the derivatives are one-unambiguous.

Similarly, we have

Proposition 20. *(1) For a one-unambiguous expression E and a symbol a, $\partial_a(E)$ can be computed as follows:*

$$\begin{aligned}
\partial_a(\emptyset) &= \partial_a(\varepsilon) = \emptyset \\
\partial_a(b) &= \begin{cases} \{\varepsilon\}, \text{ if } b = a \\ \emptyset, \quad \text{otherwise} \end{cases} \\
\partial_a(F+G) &= \begin{cases} \partial_a(F), \text{ if } a \in first(F) \\ \partial_a(G), \text{ if } a \in first(G) \\ \emptyset, \quad \text{otherwise} \end{cases} \\
\partial_a(FG) &= \begin{cases} \partial_a(F)G, \text{ if } a \in first(F) \\ \partial_a(G), \quad \text{if } a \in first(G) \text{ and } \varepsilon \in L(F) \\ \emptyset, \quad \text{otherwise} \end{cases} \\
\partial_a(F^*) &= \partial_a(F)F^*
\end{aligned}$$

(2) For a one-unambiguous expression E in SNF, the partial derivatives of E with respect to words can be computed by application of the equations in (1).

It is known [6] that derivatives of a one-unambiguous expression in SNF are one-unambiguous and in SNF. Now we have the following closure property from Proposition 20.

Corollary 8. *Partial derivatives of a one-unambiguous expression in SNF are one-unambiguous and in SNF.*

Example 5. Let E_1 be the one from Example 1, $E_1 = (ab(c+\varepsilon))^*$, and $E = a + (b^* + c)^*$. E_1 and E are one-unambiguous, with E_1 in SNF and E not.
(1) $a^{-1}(E_1) = b(c+\varepsilon)(ab(c+\varepsilon))^* = r_1$,
$(ab)^{-1}(E_1) = b^{-1}(r_1) = (c+\varepsilon)(ab(c+\varepsilon))^* = r_2$,
$(aba)^{-1}(E_1) = a^{-1}(r_2) = r_1$,
$(abc)^{-1}(E_1) = c^{-1}(r_2) = E_1$.
(2) $a^{-1}(E) = \varepsilon$,
$b^{-1}(E) = b^{-1}(b^* + c)(b^* + c)^* = b^*(b^* + c)^* = r_1$,
$\overline{r_1} = b_1^*(b_2^* + c_1)^*$, $b_1b_1, b_1b_2 \in L(\overline{r_1})$, so r_1 is not one-unambiguous.
$(bb)^{-1}(E) = b^{-1}(r_1) = r_1 + r_1$,
$\partial_b(E) = \{\varepsilon\}$,
$\partial_b(E) = \{r_1\}$,
$\partial_{bb}(E) = \{r_1\}$.
(3) $\partial_a(E_1) = \{b(c+\varepsilon)E_1\} = \{r_1\}$,
$\partial_b(r_1) = \{(c+\varepsilon)E_1\} = \{r_2\}$,
$\partial_a(r_2) = \{r_1\}$,
$\partial_c(r_2) = \{E_1\}$.

As is shown in Example 5(2), for the expression E which is not in SNF, $b^{-1}(E)$ is not one-unambiguous, then the computation of derivatives of E uses the original definition.

The above new constructions both are useful in practice for one-unambiguous expressions in SNF and provide the basis of the results below.

When an expression E is one-unambiguous, we already have that $\partial_u(E)$ is either $\emptyset$ or a singleton set, as shown in Corollary 7. Furthermore, if E is also in star normal form, then from Proposition 19 and Proposition 20 it is easy to see

Proposition 21. *For a one-unambiguous expression E in SNF, $\partial_u(E) = \emptyset$ iff $u^{-1}(E) = \emptyset$, and if $\partial_u(E) \neq \emptyset$, $\partial_u(E) = \{u^{-1}(E)\}$, for any word u.*

The above proposition sets up a one-one correspondence between derivatives and partial derivatives of a one-unambiguous expression in SNF. Then, Proposition 22 shows the relation among various sets of derivatives and partial derivatives for one-unambiguous expressions in SNF.

Proposition 22. *For a one-unambiguous expression E in SNF, we have*
(1) the set $D(E) = \{w^{-1}(E) \mid w \in \Sigma^*\}$ of derivatives of E is finite,
(2) $D(E) = PD(E)$,
(3) $\|E\| + 1 \geq |D(E)| = |DD(E)| = |PD(E)| \geq |D_1(E)| \geq |D_0(E)|$.

Proof. (1) The finiteness of $D(E)$ follows from Proposition 21.

(2) $D(E) = PD(E)$ follows from Proposition 21.

(3) The equalities and inequalities then follow from the above and $D_1(E) = D(E)/_{\sim_{aci}}$ and Proposition 1. □

The bound of $D(E)$ is worst case optimal, one example is the expression $E = abc$, for which $D(E) = \{abc, bc, c, \varepsilon\}$. Note the finiteness of $D(E)$ is also directly proved by induction in [12]. Here it is a result of Proposition 21.

If E is one-unambiguous but not in SNF, then the above correspondence between derivatives and partial derivatives does not exist (see an example in Example 5(2), where $\partial_{bb}(E) \neq \{(bb)^{-1}(E)\}$). Furthermore in this case the derivative of E with respect to a word w may not necessarily be one-unambiguous, so the number of derivatives of E with respect to words prefixed by w may be infinite.

For one-unambiguous expressions in SNF, since $D(E)$ is finite, another construction of DFA, similar to Brzozowski automaton, simply can be

$$M_{\mathrm{d}}(E) = (D(E), \Sigma, \delta, E, \{p \in D(E) \mid \varepsilon \in L(p)\}),$$

where $\delta(p, a) = a^{-1}(p)$, for any $p \in D(E), a \in \Sigma$.

An example is shown in Figure 1(b), where $M_{\mathrm{d}}(E_1)$ is the same as $M_{\mathrm{d_1}}(E_1)$.

Theorem 1. *For a one-unambiguous expression E in SNF, we have*
(1) $M_{\mathrm{pd}}(E) = M_{\mathrm{d}}(E)$,
(2) $M_{\mathrm{d_1}}(E) \simeq M_{\mathrm{d}}(E)/_{\sim_{aci}}$,
(3) $M_{\mathrm{d_1}}(E) \simeq M_{\mathrm{pd}}(E)/_{\sim_{aci}}$.

Proof. (1) $M_{\mathrm{pd}}(E) = M_{\mathrm{d}}(E)$ follows from Proposition 22.

(2) Since $D_1(E) = D(E)/_{\sim_{aci}}$, we only need to show $\sim_{aci}$ is right invariant w.r.t. $M_{\mathrm{d}}(E)$ as follows. It is easy to know that for any $p, q \in D(E)$, if $p \sim_{aci} q$ then $\varepsilon \in L(p) \Leftrightarrow \varepsilon \in L(q)$. We also show that if $p \sim_{aci} q$ then $a^{-1}(p) \sim_{aci} a^{-1}(q)$ for any $a \in \Sigma$. This can be done by induction on p. We consider the case of $p = E_1 + E_2$ here. In this case, since $p \sim_{aci} q$, q can be one of the two forms: 1. $E_1' + E_2'$, 2. $E_2' + E_1'$, where $E_i \sim_{aci} E_i', i = 1, 2$. Suppose $q = E_1' + E_2'$. Then, $a^{-1}(E_1 + E_2) = a^{-1}(E_1) + a^{-1}(E_2) \sim_{aci} a^{-1}(E_1') + a^{-1}(E_2') = a^{-1}(E_1' + E_2')$. The same applies to $q = E_2' + E_1'$. Other cases of p are similar.

(3) $M_{\mathrm{d_1}}(E) \simeq M_{\mathrm{pd}}(E)/_{\sim_{aci}}$ follows from (1) and (2). □

Corollary 9. *There is a one-unambiguous expression E in SNF, such that $M_{\mathrm{d}}(E)$ is larger than $M_{\mathrm{d_1}}(E)$.*

Proof. It suffices to show $D(E)$ may contain similar derivatives. Let $E = a(b + c) + d(c + b)$, then $a^{-1}(E) = b + c, d^{-1}(E) = c + b$, so $a^{-1}(E)$ and $d^{-1}(E)$ are similar. □

$M_{\mathrm{d}}(E)$ provides another construction of deterministic automaton by derivatives, and exists when a one-unambiguous expression E is in SNF. Corollary 9 shows that $M_{\mathrm{d}}(E)$ and $M_{\mathrm{d_1}}(E)$ can be different.

The sets of derivatives and partial derivatives and their relations discussed in the previous sections and this section are summarized in Table 1.

Table 1. Summary of sets of (partial) derivatives. NSNF–Non SNF, $w = \|E\|$

Regular expression E		$D(E), D_1(E), D_0(E), DD(E), PD(E)$ and their relations
one-unambiguous	SNF	$D(E) = PD(E) \sim DD(E)$, $D_1(E) = D(E)/_{\sim_{aci}}$, $D_0(E) \sim D_1(E)/_{\approx}$ $w+1 \geq \|D(E)\| = \|DD(E)\| = \|PD(E)\| \geq \|D_1(E)\| \geq \|D_0(E)\|$
	NSNF	$DD(E) \sim PD(E)$, $D_0(E) \sim D_1(E)/_{\approx}$ $w+1 \geq \|DD(E)\| = \|PD(E)\| \geq \|D_0(E)\|$, $\|D_1(E)\| \geq \|D_0(E)\|$
other		$D_0(E) \sim D_1(E)/_{\approx}$ $2^{w+1} \geq \|DD(E)\| \geq \|D_0(E)\|$, $w+1 \geq \|PD(E)\|$, $\|D_1(E)\| \geq \|D_0(E)\|$

7 Concluding remarks

1. For a one-unambiguous expression E in SNF, since $M_{\mathrm{d}}(E)$ is equal to $M_{\mathrm{pd}}(E)$, we obtain a way to reduce the size of $M_{\mathrm{pd}}(E)$ by using dissimilar states in the building of the automaton. The resulting automaton is equal to $M_{\mathrm{d_1}}(E)$.

2. The Brzozowski automaton $M_{\mathrm{d_1}}(E)$ may not be minimal. If we consider equality of derivatives, and use $D_0(E)$ as the set of states, the resulting DFA is minimal [7]. However, equality test has PSPACE time complexity for regular expressions, and is in quadratic time for one-unambiguous expressions [12].

3. Usually one would consider that the Brzozowski automaton is the determinisation (by the subset construction) of $M_{\mathrm{pd}}(E)$ (e. g., [19]). This is however imprecise. Actually $M_{\mathrm{dd}}(E)$ is exactly converted by subset construction from $M_{\mathrm{pd}}(E)$ [1]. However, from Theorem 1, we know that $M_{\mathrm{d_1}}(E)$ and $M_{\mathrm{pd}}(E)$ can be different, which proves $M_{\mathrm{d_1}}(E)$ is not the determinisation of $M_{\mathrm{pd}}(E)$.

The paper discussed derivatives, partial derivatives and automata in the case of expressions in SNF and of one-unambiguous expressions. It showed that if an expression E is in SNF, then $(wx)^{-1}(\overline{E})$ is either $\emptyset$ or unique for all words w, which is a stronger property than Berry and Sethi's [2]. For a regular expression in SNF it presented several conditions for the quotient or isomorphism relation between the equation and follow automata. The paper have given a figure, though probably incomplete yet, of various relations that are possible between the two automata. Star normal form also makes sense for one-unambiguous expressions, in which case the expressions bear more good properties; For example, a simple construction of the Brzozowski automaton can be get, and the paper showed that the resulting automaton is equal to the equation automaton. It showed that the equation and follow automata can be computed in linear time for one-unambiguous expressions.

Several problems can be investigated as future work. For example, as mentioned in a previous section, whether there are some more conditions for the relation between $M_{\mathrm{pd}}(E)$ and $M_{\mathrm{f}}(E)$ remains as a further research.

References

1. V. Antimirov, Partial derivatives of regular expressions and finite automaton constructions, Theoretical Computer Science 155 (1996) 291–319.
2. G. Berry, R. Sethi, From regular expressions to deterministic automata, Theoretical Computer Science 48 (1986) 117–126.
3. R. Book, S. Even, S. Greibach, and G. Ott, Ambiguity in graphs and expressions, IEEE Transactions on Computers, C-20(2):149–153, 1971.
4. A. Bruggemann-Klein, Regular expressions into finite automata, Theoretical Computer Science 120 (1993) 197–213.
5. A. Bruggemann-Klein and D. Wood, Deterministic regular languages, STACS 92, LNCS 577, Springer, 1992, 173–184.
6. A. Bruggemann-Klein and D. Wood, One-unambiguous regular languages, Information and Computation, 142(2):182–206, 1998.

7. J. A. Brzozowski, Derivatives of regular expressions, J. ACM 11(4):481–494, 1964.
8. J.-M. Champarnaud, F. Nicart, and D. Ziadi, Computing the follow automaton of an expression, CIAA 2004, LNCS 3317, Springer, 2005, 90–101.
9. J.-M. Champarnaud, D. Ziadi, Canonical derivatives, partial derivatives and finite automaton constructions, Theoretical Computer Science 289 (2002) 137–163.
10. J.-M. Champarnaud, F. Ouardi, and D. Ziadi, Normalized expressions and finite automata, International J. of Algebra and Computation, 17(1), 141–154, 2007.
11. C.-H. Chang and R. Page, From regular expressions to DFA's using compressed NFA's, Theoretical Computer Science, 178(1-2):1–36, 1997.
12. H. Chen, L. Chen, Inclusion test algorithms for one-unambiguous regular expressions, ICTAC 2008, LNCS 5160, Springer, 2008, 96–110.
13. C. Chitic and D. Rosu, On validation of XML streams using finite state machines, WebDB 2004, 85–90.
14. D. Giammarresi, R. Montalbano, D. Wood, Block-deterministic regular languages, ICTCS 2001, LNCS 2202, Springer, 2001, 184–196.
15. V. M. Glushkov, The abstract theory of automata, Russian Math. Surveys 16 (1961) 1–53.
16. L. Ilie and S. Yu, Follow automata, Information and Computation, 186(1):146–162.
17. P. Kilpelainen and R. Tuhkanen, One-unambiguity of regular expressions with numerical occurrence indicators, Information and Computation 205(6):890–916.
18. A. Khorsi, F. Ouardi, and D. Ziadi, Fast equation automaton computation, International Journal of Discrete Algorithms, Vol. 6, No. 3, pp 433-448 (2008).
19. S. Lombardy and J. Sakarovitch, Derivatives of rational expressions with multiplicity, Theoretical Computer Science 332(1-3): 141–177, 2005.
20. R. McNaughton, H. Yamada, Regular expressions and state graphs for automata, IEEE Trans. on Electronic Computers 9 (1) (1960) 39–47.
21. B. G. Mirkin. An algorithm for constructing a base in a language of regular expressions. Engineering Cybernetics, 5:110–16, 1966.
22. J.-L. Ponty, D. Ziadi and J.-M. Champarnaud, A new Quadratic Algorithm to convert a Regular Expression into an Automaton, WIA'96, LNCS 1260, 1997, 109–119.
23. S. Yu, Regular Languages, in: G. Rozenberg, A. Salomaa, eds., Handbook of Formal Languages, Vol. I, Springer-Verlag, Berlin, 1997, 41–110.