Derivatives and Finite Automata of Expressions in Star Normal Form

Haiming Chen[1(✉)] and Ping Lu[2]

[1] State Key Laboratory of Computer Science,
Institute of Software Chinese Academy of Sciences, Beijing 100190, China
chm@ios.ac.cn
[2] BDBC, Beihang University, Beijing 100191, China
luping@buaa.edu.cn

Abstract. This paper studies derivatives and automata for expressions in star normal form as defined by Brüggemann-Klein. For an expression in star normal form, the paper shows that the derivatives are either $\emptyset$ or unique, while in general Berry and Sethi's result shows the derivatives are either $\emptyset$ or similar. It is known that the partial derivative automaton and the follow automaton are two small automata, each of which is a quotient of the position automaton. For the relation between the partial derivative 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. Our work shows that there are different conditions under which the relation of the two automata can be different.

Keywords: Regular expressions · Finite automata · Derivatives · Partial derivatives · Star normal form

1 Introduction

Finite automata are basic for efficient implementation and application of regular expressions. Derivatives are a fundamental concept for regular expressions and a useful tool to study automata construction from regular expressions. This paper studies derivatives and automata of regular expressions in star normal form, defined by Brüggemann-Klein [3]. It is known that every regular expression can be transformed into star normal form in linear time [3], and several algorithms depend on star normal form (e. g., [3,8]).

Derivatives of regular expressions were introduced by Brzozowski [5]. The notion was generalized to partial derivatives by Antimirov [1]. There has been no result about derivatives particular for expressions in star normal form. Among the many constructions of ϵ-free non-deterministic finite automata (NFA) from regular expressions, we consider *position automata* proposed separately by Glushkov [10] and McNaughton and Yamada [12], *partial derivative* or *equation*

Work supported by the National Natural Science Foundation of China under Grant No. 61472405.

F. Drewes et al. (Eds.): LATA 2017, LNCS 10168, pp. 236–248, 2017.
DOI: 10.1007/978-3-319-53733-7_17

automata using partial derivatives [1], and *follow automata* proposed by Ilie and Yu [11]. The position automaton has size at most quadratic and can be computed in quadratic time [3,9,14]. Berry and Sethi [2] showed a natural connection between the position automaton and the derivatives. The partial derivative automaton has also been proved to be equivalent to the automaton constructed from the prebase [13]. Champarnaud and Ziadi [7] proposed a quadratic algorithm for computing the partial derivative automata which improved very much the original algorithm [1], and proved that the partial derivative automaton is a quotient of the position automaton. Ilie and Yu [11] proposed a simplified proof of the result. Lombardy and Sakarovitch [15] gave another proof in the more general setting of expressions with multiplicity which applies to present Boolean case. Recently Ilie and Yu [11] 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 paper first shows that, for an expression in star normal form, the derivatives of the marked expression (see Sect. 2 for the explanation of marked expression) with respect to word of the form wa for any word w and a fixed symbol a are either $\emptyset$ or unique, while Berry and Sethi's result [2] establishes that in general the above derivatives are either $\emptyset$ or similar. This uniqueness of derivatives is of course an attractive property.

The paper then discusses the relation between the partial derivative and follow automata. It has been known that both the partial derivative and follow automata are quotients of the position automaton. The question is what is the relation between the first two automata. In [11] Ilie and Yu compared some examples and stated that "a more rigorous comparison" between the automata "should be done" but "seems difficult". Champarnaud et al. [6] gave a condition ("normalized" regular expressions) under which the partial derivative automaton is a quotient of the follow automaton[1]. The paper gives several conditions for the following relations between the two automata: (1) the partial derivative automaton is a quotient of the follow automaton, (2) the converse, and (3) 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. Then based on the structure of expressions, 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 *CONC* condition (see Sect. 4), the partial derivative automaton is a quotient of the follow automaton. Compared with the work in [6], the *CONC* condition is more general (meaning that it allows more expressions than normalized regular expressions), and captures more adequately the nature of expressions for which the resulting partial derivative and follow automata

[1] This quotient result, however, is not given in [6]. In [6] the main theorem (Theorem 4, p.11) states for a "normalized" regular expression, the size of the partial derivative automaton is smaller than the size of the follow automaton.

retain the above quotient relation. For example, none of the expressions given in Example 37 are normalized regular expressions, while they all satisfy *CONC* condition, and for each of the expressions the partial derivative automaton is a quotient of the follow automaton. See Sect. 4 for a discussion. We further present conditions for some special situations, in which the two automata are isomorphic or the follow automaton is a quotient of the partial derivative automaton. Since regular expressions can be transformed to star normal form in linear time, we can easily get the smaller automaton when one of the above conditions is satisfied.

Section 2 introduces basic notations and notions. Derivatives for expressions in star normal form are considered in Sect. 3. Section 4 focuses on the relation of partial derivative and follow automata. Section 5 gives concluding remarks.

2 Preliminaries

We assume the reader to be familiar with basic regular language and automata theory, e.g., from [16]. We introduce here only some notations and notions used later in the paper.

Let Σ be an alphabet of symbols. 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 the union E_1+E_2, the concatenation E_1E_2, or the star E_1^* for regular expressions E_1 and E_2. 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 rules $E+\emptyset = \emptyset+E = E, E\emptyset = \emptyset E = \emptyset$, and $E\varepsilon = \varepsilon E = E$ (rules-$\emptyset\varepsilon$) hold in the paper.

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 $(a + b)^*ab(a + b)$. The 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 $\bar{\cdot}$ adds or drops subscripts.

For an expression E over Σ, we define the following sets: $first(E) = \{a \mid aw \in L(E), a \in \Sigma, w \in \Sigma^*\}, last(E) = \{a \mid wa \in L(E), w \in \Sigma^*, a \in \Sigma\}, follow(E, a) = \{b \mid uabv \in L(E), u, v \in \Sigma^*, b \in \Sigma\}$ for $a \in \Sigma$.

Define $followlast(E) = \{b \mid vbw \in L(E), v \in L(E), v \neq \varepsilon, b \in \Sigma, w \in \Sigma^*\}$. 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 [3].

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 $L(M)$.

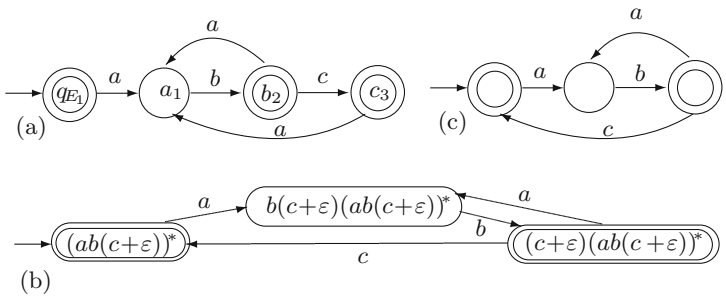

Fig. 1. (a) $M_{\mathrm{pos}}(E_1)$, (b) $M_{\mathrm{pd}}(E_1)$, and (c) $M_{\mathrm{f}}(E_1)$, corresponding to $E_1 = (ab(c+\varepsilon))^*$.

Let $\equiv \subseteq Q \times Q$ be an equivalence relation. 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 $p_1 \equiv q_1$ for $p_1 \in \delta(p,a), q_1 \in \delta(q,a)$. If $\equiv$ is right invariant, then we can define a quotient automaton $M/_{\equiv}$ in the usual way.

The position automaton was introduced independently by Glushkov [10] and McNaughton and Yamada [12]. The *position automaton* of E is $M_{\mathrm{pos}}(E) = (Q_{\mathrm{pos}}, \Sigma, \delta_{\mathrm{pos}}, q_E, F_{\mathrm{pos}})$, where $Q_{\mathrm{pos}} = \Sigma_{\overline{E}} \cup \{q_E\}$, $\delta_{\mathrm{pos}}(q_E, a) = \{x \mid x \in first(\overline{E}), \overline{x} = a\}$ for $a \in \Sigma$, $\delta_{\mathrm{pos}}(x,a) = \{y \mid y \in follow(\overline{E}, x), \overline{y} = a\}$ for $x \in \Sigma_{\overline{E}}$ and $a \in \Sigma$, $F_{\mathrm{pos}} = last(\overline{E}) \cup \{q_E\}$ if $\lambda(E) = \varepsilon$, or $last(\overline{E})$ otherwise.

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 $M_{\mathrm{pos}}(E_1)$ for the regular expression $E_1 = (ab(c+\varepsilon))^*$ is shown in Fig. 1(a).

As shown by Glushkov [10] and McNaughton and Yamada [12], $L(M_{\mathrm{pos}}(E)) = L(E)$. $M_{\mathrm{pos}}(E)$ can be computed in quadratic time [3,9,14].

Below we introduce deravatives.

Definition 2 (Brzozowski [5]). *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:*

$$a^{-1}(\emptyset) = a^{-1}(\varepsilon) = \emptyset$$
$$a^{-1}(b) = \varepsilon \text{ if } b = a,\ \emptyset \text{ otherwise}$$
$$a^{-1}(F+G) = a^{-1}(F) + a^{-1}(G)$$
$$a^{-1}(FG) = a^{-1}(F)G + a^{-1}(G) \text{ if } \lambda(F) = \varepsilon,\ a^{-1}(F)G \text{ otherwise}$$
$$a^{-1}(F^*) = a^{-1}(F)F^*$$

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

Definition 3 (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:*[2]

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

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 $L(\partial_w(E)) = \bigcup_{p\in\partial_w(E)} L(p)$.

It is proved in [1] that the cardinality of the set $PD(E) = \cup_{w\in\Sigma^*}\partial_w(E)$ of all partial derivatives of a regular expression E is less than or equal to $\|E\| + 1$.

The *partial derivative* or *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 Fig. 1(b).

It is proved that for a regular expression, the partial derivative automaton is a quotient of the position automaton [7,11]. Another proof is given by Lombardy and Sakarovitch [15], which is in the more general setting of expressions with multiplicity but still applies to present case (multiplicities over the Boolean semiring).

Expressions with distinct symbols are called linear. For any expression E, $\overline{E}$ is the linearized version of E. For linear expressions from Brzozowski [5] and Berry and Sethi [2] the following fact is easily derived.

Proposition 4. *Let E be linear. Given $a \in \Sigma_E$, for all words w,*

1. If $E = E_1 + E_2$, then

$$(wa)^{-1}(E_1 + E_2) = \begin{cases} (wa)^{-1}(E_1) \text{ if } a \in \Sigma_{E_1} \\ (wa)^{-1}(E_2) \text{ if } a \in \Sigma_{E_2} \end{cases} \tag{1}$$

2. If $E = E_1E_2$, then

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

3 Derivatives of Expressions in SNF

Two regular expressions E_1 and E_2 which reduce to the same expression using associativity, commutativity, and idempotence of $+$ are called *ACI-similar* [5],

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

which is denoted by $E_1 \sim_{aci} E_2$. 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 5. *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 $(wx)^{-1}(\overline{E_1})$, and the inductive hypothesis applies to it. The case for x in $\overline{E_2}$ follows in the same way.
2. $\overline{E} = \overline{E_1}\,\overline{E_2}$. 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 [5] 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 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 $wx = a_1a_2 \ldots a_t$. We can suppose $v_1x = a_{r_1} \ldots a_t, v_2x = a_{r_2} \ldots a_{r_1} \ldots a_t, 1 \leq r_2 < r_1 \leq t$. 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 $a_{r_2} \ldots a_{r_1} \ldots a_t w_1 \in L(\overline{E_1})$. 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 $\varepsilon \in L((a_n \ldots a_{r_1-1})^{-1}(\overline{E_1}))$ for some $n < a_{r_1-1}$. In either case, we have $a_{r_1-1} \in last(\overline{E_1})$. Note the symbols and positions are in one-one correspondence for $\overline{E}$. Therefore E is not in SNF, which is a contradiction.

If $v_1 = v_2$, then $v_1x = v_2x = a_{r_1} \ldots a_t, 2 < r_1 \leq t$. Similarly, a careful analysis on $(wx)^{-1}(\overline{E})$ shows that there must be $\varepsilon \in L((a_{n_1} \ldots a_i)^{-1}(\overline{E_1})), \varepsilon \in L((a_{n_2} \ldots a_i)^{-1}(\overline{E_1}))$ and $\varepsilon \in L((a_{n_3} \ldots a_{n_1-1})^{-1}(\overline{E_1})), n_2 < n_1 \leq i \leq r_1 - 1, n_3 < n_1$. 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. □

This uniqueness of derivatives is of course an attractive property. There have been several work relying on finding a unique representative for the set of non-null $(wx)^{-1}(\overline{E})$ [7,11]. If E is in SNF, then $(wx)^{-1}(\overline{E})$ is already unique.

Corollary 6. *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 5 above, it follows

Corollary 7. *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 Partial Derivative and Follow Automata

The *follow automaton* $M_{\mathrm{f}}(E)$ was introduced by Ilie and Yu [11]. It is constructed by eliminating ε-transitions from an ε-automaton defined in [11]. We do not present the construction in detail here. An example is shown in Fig. 1(c). What is important here is the following.

Define the equivalence $\equiv_f \subseteq Q_{\mathrm{pos}}^2$ 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. $M_{\mathrm{pos}}(E)$. Define $M_1 \simeq M_2$ if M_1 and M_2 are isomorphic. It is known that

Proposition 8 [11]. $M_{\mathrm{f}}(E) \simeq M_{\mathrm{pos}}(E)/_{\equiv_f}$.

As we have mentioned, it is well-known that the partial derivative automaton is a quotient of the position automaton [7,11,15]. Here it is presented following [11]. 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 an SNF expression E, $C_x(\overline{E})$ is already unique. For general expressions assume that we find a proper representative for each $C_x(\overline{E})$ [7,11]. Define the equivalence $=_c \subseteq Q_{\mathrm{pos}}^2$ by $x_1 =_c x_2$ iff $C_{x_1}(\overline{E}) = C_{x_2}(\overline{E})$. Define the equivalence $\equiv_c \subseteq Q_{\mathrm{pos}}^2$ 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. $M_{\mathrm{pos}}(E)$. It is known that

Proposition 9. *(1)* $M_{\mathrm{pd}}(E) \simeq M_{\mathrm{pos}}(E)/_{\equiv_c}$; *(2)* $\overline{M_{\mathrm{pd}}(\overline{E})} \simeq M_{\mathrm{pos}}(E)/_{=_c}$.

From Propositions 8 and 9 both $M_{\mathrm{pd}}(E)$ and $M_{\mathrm{f}}(E)$ are always smaller than or equal to $M_{\mathrm{pos}}(E)$. However, for the relation between $M_{\mathrm{pd}}(E)$ and $M_{\mathrm{f}}(E)$, Ilie and Yu [11] compared some examples and showed that it is difficult to give a theoretical analysis. Here we try to do so.

First we give characterizations of the different relations between the two automata. It is easy to see the following:

Lemma 10. *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 10 and the above definitions of the equivalence relations, the following are implied

Lemma 11. *(1)* $=_c \subseteq \equiv_f$; *(2)* $=_c \subseteq \equiv_c$.

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

Proposition 12. *For an expression E, we have $=_c \;=\; \equiv_f$ iff $\forall a, b \in Q_{\mathrm{pos}}$, $first(C_a(\overline{E})) = first(C_b(\overline{E})) \wedge \lambda(C_a(\overline{E})) = \lambda(C_b(\overline{E})) \Rightarrow C_a(\overline{E}) = C_b(\overline{E})$.*

Similarly the following are other characterizations.

Proposition 13. *For an expression E, we have $=_c \;=\; \equiv_c$ iff $\forall a, b \in Q_{\mathrm{pos}}$, $\overline{C_a(\overline{E})} = \overline{C_b(\overline{E})} \Rightarrow C_a(\overline{E}) = C_b(\overline{E})$.*

Proposition 14. *For an expression E, we have $\equiv_c \;=\; \equiv_f$ iff $\forall a, b \in Q_{\mathrm{pos}}$, $\overline{C_a(\overline{E})} = \overline{C_b(\overline{E})} \Leftrightarrow first(C_a(\overline{E})) = first(C_b(\overline{E})) \wedge \lambda(C_a(\overline{E})) = \lambda(C_b(\overline{E}))$.*

On the other hand, from Propositions 8, 9 and Lemma 11 it follows

Theorem 15. *For an expression E,*

(1) if $=_c \;=\; \equiv_f$, then $M_{\mathrm{pd}}(E)$ is a quotient of $M_{\mathrm{f}}(E)$, $\overline{M_{\mathrm{pd}}(\overline{E})} \simeq M_{\mathrm{f}}(E)$; and
(2) if $=_c \;=\; \equiv_c$, then $M_{\mathrm{f}}(E)$ is a quotient of $M_{\mathrm{pd}}(E)$, $\overline{M_{\mathrm{pd}}(\overline{E})} \simeq M_{\mathrm{pd}}(E)$; and
(3) if $\equiv_c \;=\; \equiv_f$, then $M_{\mathrm{pd}}(E) \simeq M_{\mathrm{f}}(E)$.

Example 16. Let $E_1 = aa^* + ba^*, E_2 = (a^* + \varepsilon)a^*a^*, E_3 = a^*$, one can verify that $M_{\mathrm{pd}}(E_1)$ is a quotient of $M_{\mathrm{f}}(E_1)$, $M_{\mathrm{f}}(E_2)$ is a quotient of $M_{\mathrm{pd}}(E_2)$, and $M_{\mathrm{pd}}(E_3) \simeq M_{\mathrm{f}}(E_3)$.

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:

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

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

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

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

Lemma 17. *For $b \in \Sigma_{\overline{E}}$, if $follow(\overline{E}, b) = \emptyset$ then $b \in last(\overline{E})$.*

Lemma 18. *For $b \in \Sigma_{\overline{E}}$, $follow(\overline{E}, b) = \emptyset$ iff $\forall w \in \Sigma^*_{\overline{E}}$, $(wb)^{-1}(\overline{E}) = \emptyset$ or $(wb)^{-1}(\overline{E}) = \varepsilon$.*

Lemma 19. *For $\overline{E} = \overline{F} + \overline{G}$, $a \in \Sigma_{\overline{F}}$ and $b \in \Sigma_{\overline{G}}$, $a \equiv_f b \Leftrightarrow a =_c b$.*

Definition 20. *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 $F + \varepsilon$ or $\varepsilon + F$ where $\lambda(F) = \varepsilon$. Obviously if $E = FG$ then E is leftmost ε-reduced iff F is leftmost ε-reduced.*

The expressions $a + \varepsilon$, $(a^* + \varepsilon)^*$, $a + (a^* + \varepsilon)b$, $b^*(a^* + \varepsilon)$, $(a + \varepsilon) + b^*$ are leftmost ε-reduced, while $a^* + \varepsilon$, $(a + \varepsilon) + \varepsilon$, $(a^* + \varepsilon)b^*$, $(\varepsilon + a^*)b$, $(a + b^*) + \varepsilon$ are not leftmost ε-reduced.

Definition 21. *For an expression E, and $b \in \Sigma_{\overline{E}}$, we denote $\psi_1(E, b)$ the following condition: $b \in last(\overline{E}) \Leftrightarrow \varepsilon \in L(\overline{E})$; and denote $\psi_2(E, b)$ the following condition: $first(\overline{E}) = follow(\overline{E}, b)$.*

Lemma 22. *The following are equivalent statements.*

(1) $q_E \equiv_f b$;
(2) $\psi_1(E, b)$ and $\psi_2(E, b)$;
(3) $\forall w \in \Sigma^*_{\overline{E}}$, if $(wb)^{-1}(\overline{E}) \neq \emptyset$ then $L((wb)^{-1}(\overline{E})) = L(\overline{E})$.

Definition 23. *For an expression E, we call the following the emptiness condition of E: If $le(\overline{E}) = \overline{F}^*, b \in last(\overline{F})$, and $first(\overline{F}^*) = follow(\overline{F}^*, b)$, then $followlast(\overline{F}) \cap first(\overline{F}) = \emptyset$.*

Lemma 24. *For an expression E and $b \in \Sigma_{\overline{E}}$, if $\psi_1(E, b)$, $\psi_2(E, b)$, E is leftmost ε-reduced, and satisfies the emptiness condition, then $\forall w \in \Sigma^*_{\overline{E}}$, if $(wb)^{-1}(\overline{E}) \neq \emptyset$ then $(wb)^{-1}(\overline{E}) = \overline{E}$.*

From the proof of the above lemma, it follows

Corollary 25. *For an expression E and $b \in \Sigma_{\overline{E}}$, if $\psi_1(E, b)$, $\psi_2(E, b)$, E is leftmost ε-reduced, and satisfies the emptiness condition, 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 the same conditions as for E.*

It is easy to see that $q_E =_c b$ equals the statement: $\forall w \in \Sigma^*_{\overline{E}}$, if $(wb)^{-1}(\overline{E}) \neq \emptyset$ then $(wb)^{-1}(\overline{E}) = \overline{E}$. Then from Lemmas 11 and 22 we have the following:

Lemma 26. *Given $b \in \Sigma_{\overline{E}}$, if $\forall w \in \Sigma^*_{\overline{E}}$, and whenever $(wb)^{-1}(\overline{E}) \neq \emptyset$ we have $(wb)^{-1}(\overline{E}) = \overline{E}$, then $\psi_1(\overline{E}, b)$ and $\psi_2(\overline{E}, b)$.*

Definition 27. *For any starred subexpression F^* of an expression E, we call the following the equivalence condition of E: If $a, b \in last(\overline{F})$ and $follow(\overline{F}^*, a) = follow(\overline{F}^*, b)$, then $follow(\overline{F}, a) = follow(\overline{F}, b)$.*

For an SNF expression, we have

Lemma 28. *Suppose F^* is in SNF. If $a, b \in last(\overline{F})$ and $follow(\overline{F}^*, a) = follow(\overline{F}^*, b)$, then $follow(\overline{F}, a) = follow(\overline{F}, b)$.*

Definition 29. *For an expression E, we call the following the CONC (Concatenation) condition of E: If $\psi_1(E,b)$ and $\psi_2(E,b)$ for some $b \in \Sigma_{\overline{E}}$ then E is leftmost ε-reduced and satisfies the emptiness condition; For any subexpression $\overline{FG}$ of $\overline{E}$, if $follow(\overline{F},a) = \emptyset$, $\psi_1(G,b)$ and $\psi_2(G,b)$ for some $a \in \Sigma_{\overline{F}}$ and $b \in \Sigma_{\overline{G}}$ then G is leftmost ε-reduced and satisfies the emptiness condition.*

The significance of the *CONC* condition can be seen from the following two lemmas.

Lemma 30. *For $\overline{E} = \overline{FG}$, $a \in \Sigma_{\overline{F}}$ and $b \in \Sigma_{\overline{G}}$, $a \equiv_f b$ iff $follow(\overline{F},a) = \emptyset$, $\psi_1(G,b)$ and $\psi_2(G,b)$.*

Lemma 31. *For $\overline{E} = \overline{FG}$, $a \in \Sigma_{\overline{F}}$ and $b \in \Sigma_{\overline{G}}$, if $follow(\overline{F},a) = \emptyset$, $\psi_1(G,b)$, $\psi_2(G,b)$, G is leftmost ε-reduced and satisfies the emptiness condition, then $a =_c b$.*

From Lemmas 11 and 30 it follows

Corollary 32. *For $\overline{E} = \overline{FG}$, $a \in \Sigma_{\overline{F}}$ and $b \in \Sigma_{\overline{G}}$, if $a =_c b$ then $follow(\overline{F},a) = \emptyset$, $\psi_1(G,b)$ and $\psi_2(G,b)$.*

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

Theorem 33. *For an expression E satisfying CONC and equivalence conditions, we have $=_c = \equiv_f$.*

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 $b_3 \notin last(\overline{E})$.

According to Lemma 28, we have

Corollary 34. *For an SNF expression E satisfying CONC condition, we have $=_c = \equiv_f$.*

Corollary 35. *For an expression E satisfying CONC and equivalence conditions, $\overline{M_{\mathrm{pd}}(\overline{E})} \simeq M_{\mathrm{f}}(E)$, and $M_{\mathrm{pd}}(E)$ is a quotient of $M_{\mathrm{f}}(E)$.*

Corollary 36. *For an SNF regular expression E satisfying CONC condition, $\overline{M_{\mathrm{pd}}(\overline{E})} \simeq M_{\mathrm{f}}(E)$, and $M_{\mathrm{pd}}(E)$ is a quotient of $M_{\mathrm{f}}(E)$.*

Example 37. 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 *CONC* condition. One can verify that for each E_i, $=_c = \equiv_f$, and $M_{\mathrm{pd}}(E_i)$ is a quotient of $M_{\mathrm{f}}(E_i), i = 1,2,3$.

Also, the expressions E_1 in Example 1, E_1 and E_3 in Example 16 are in SNF and satisfy *CONC* condition, and their partial derivative automata are quotients of the follow automata.

Remark. Champarnaud et al. [6] has proposed a condition called "normalized" regular expressions, which requires that in an SNF expression no subexpression

$F + \varepsilon$ with $\lambda(F) = \varepsilon$ should exist. For a "normalized" regular expression the partial derivative automaton is a quotient of the follow automaton. A "normalized" expression is of course leftmost ε-reduced, and trivially satisfies *CONC* condition. Conversely, if an SNF expression satisfies *CONC* condition, it may not necessarily be a "normalized" expression. For example, none of expressions given in Example 37 are "normalized", while they all satisfy *CONC* condition, and for each of the expressions the partial derivative automaton is a quotient of the follow automaton. As the above example hints, for many expressions that are not "normalized", the quotient relation between the partial derivative automaton and the follow automaton still exists. Actually "normalized" expressions simply forbid the occurrence of any subexpression $F + \varepsilon$ where $\lambda(F) = \varepsilon$, but *CONC* condition only forbid the occurrence of the above subexpression in some sensitive positions in an expression. Therefore "normalized" expressions impose too strong restrictions on expressions to ensure the quotient. On the other hand, *CONC* condition captures more adequately the nature of expressions for which the partial derivative automaton is a quotient of the follow automaton.

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

Condition 1. Let $E = F_1F_2 \dots F_n, F_r$ 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^*.

Theorem 38. *For a regular expression E satisfying Condition 1, we have $=_c = \equiv_f$, $=_c = \equiv_c$, and $\equiv_c = \equiv_f$.*

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.

Theorem 39. *For a regular expression E satisfying Condition 2, we have $=_c \neq \equiv_f$ and $=_c = \equiv_c$.*

Corollary 40. *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 16 and E_1 in Example 37 satisfy Condition 1, and their partial derivative and follow automata are isomorphic. The expression E_2 in Example 16 satisfies Condition 2, and its follow automaton is a quotient of the partial derivative 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

Theorem 41. *For a linear expression E, we have $=_c = \equiv_c$.*

Corollary 42. *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 37, we have $M_{\mathrm{pd}}(E_3) \simeq M_{\mathrm{f}}(E_3)$ since it is linear and from Example 37 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 [3], we can easily get the smaller automaton when one of the above conditions is satisfied. Further, it would be interesting to find some more conditions, which remains as a further research.

5 Concluding Remarks

The paper discussed derivatives and automata for expressions in SNF. 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 partial derivative and follow automata.

Several problems can be investigated as future work. For conditions based on the structure of expressions, although *CONC* condition allows more expressions, presently it is unclear whether it captures all expressions for which $M_{\mathrm{pd}}(E)$ is a quotient of $M_{\mathrm{f}}(E)$. As mentioned above, 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. Antimirov, V.: Partial derivatives of regular expressions and finite automaton constructions. Theoret. Comput. Sci. **155**, 291–319 (1996)
2. Berry, G., Sethi, R.: From regular expressions to deterministic automata. Theoret. Comput. Sci. **48**, 117–126 (1986)
3. Brüggemann-Klein, A.: Regular expressions into finite automata. Theoret. Comput. Sci. **120**, 197–213 (1993)
4. Brüggemann-Klein, A., Wood, D.: One-unambiguous regular languages. Inf. Comput. **142**(2), 182–206 (1998)
5. Brzozowski, J.A.: Derivatives of regular expressions. J. ACM (JACM) **11**(4), 481–494 (1964)
6. Champarnaud, J.-M., Ouardi, F., Ziadi, D.: Normalized expressions and finite automata. Int. J. Algebra Comput. **17**(01), 141–154 (2007)
7. Champarnaud, J.-M., Ziadi, D.: Canonical derivatives, partial derivatives and finite automaton constructions. Theoret. Comput. Sci. **289**(1), 137–163 (2002)
8. Champarnaud, J.-M., Nicart, F., Ziadi, D.: Computing the follow automaton of an expression. In: Domaratzki, M., Okhotin, A., Salomaa, K., Yu, S. (eds.) CIAA 2004. LNCS, vol. 3317, pp. 90–101. Springer, Heidelberg (2005). doi:10.1007/978-3-540-30500-2_9

9. Chia-Hsiang, C., Paige, R.: From regular expressions to DFA's using compressed NFA's. Theoret. Comput. Sci. **178**(1), 1–36 (1997)
10. Glushkov, V.M.: The abstract theory of automata. Russ. Math. Surv. **16**(5), 1 (1961)
11. Ilie, L., Yu, S.: Follow automata. Inf. Comput. **186**, 146–162 (2003)
12. McNaughton, R., Yamada, H.: Regular expressions and state graphs for automata. IEEE Trans. Electron. Comput. **1**(EC–9), 39–47 (1960)
13. Mirkin, B.G.: An algorithm for constructing a base in a language of regular expressions. Eng. Cybern. **5**, 110–116 (1966)
14. Ponty, J.-L., Ziadi, D., Champarnaud, J.-M.: A new quadratic algorithm to convert a regular expression into an automaton. In: Raymond, D., Wood, D., Yu, S. (eds.) WIA 1996. LNCS, vol. 1260, pp. 109–119. Springer, Heidelberg (1997). doi:10.1007/3-540-63174-7_9
15. Lombardy, S., Sakarovitch, J.: Derivatives of rational expressions with multiplicity. Theoret. Comput. Sci. **332**(1–3), 141–177 (2005)
16. Yu, S.: Regular languages. In: Rozenberg, G., Salomaa, A. (eds.) Handbook of Formal Languages, vol. 1, pp. 41–110. Springer, Berlin (1997)