Chiang Mai J. Sci. 2014; 41(5.2) : 1409-1418
http://epg.science.cmu.ac.th/ejournal/
Contributed Paper

Conversion of Parallel Regular Expressions to Non-deterministic Finite Automata using Partial Derivatives

Ajay Kumar and Anil Kumar Verma

Department of Computer Science and Engineering Department, Thapar University, Patiala, India.

*Author for correspndence; e-mail: ajayloura@gmail.com

Received: 18 October 2012
Accepted: 31 May 2013

ABSTRACT

Several techniques like Thompson's construction, partial derivatives, follow automata and position automata have been proposed for the conversion of regular expressions to non-deterministic finite automata. Researchers have proposed different methodologies for the conversion of parallel regular expressions to non-deterministic finite automata having the number of states $O(2^{|r|})$. The aim of the paper is to propose a new approach for the conversion of parallel regular expressions to non-deterministic finite automata. This innovative approach is the generalization of the Antimirov partial derivatives for the conversion of regular expressions to ε-free non-deterministic finite automata. The number of states of the non-deterministic finite automaton is reduced from exponential $(O(2^{|r|}))$ to polynomial $(O(m^{k+1}))$, using this novel approach.

Keywords: non-deterministic finite automata, parallel regular expression, regular expression, shuffle operator

1. Introduction

Regular expressions (*REs*) are used in many applications such as compiler design, text processor, finite state machine based testing, parallel processing, *XML* schema and as a programming tool in many scripting languages such as *PHP*, *Perl* [1-6]. Extended *REs* consist of intersection, complement and counting operators in addition to the union, Kleene closure and concatenation operators [7]. *RE* having Kleene closure (*), union (| or +), concatenation (.) and shuffle operator (III) is known as a parallel regular expression (*PRE*) [8]. Shuffle operator appears in various fields of computer science like process algebra [9], *XML* schema language *Relax NG* [2], multipoint communication [10] and concurrency of processes [8, 11-12]. Conversion of *PRE* to *RE* or finite automaton give us a way to study the serialization of concurrency [8], as well as translation of *Relax NG* to *XML* schema definition [2].

Thompson [13] proposed the first technique for the conversion of *RE* to non-deterministic finite automaton (*NFA*). Sippu and Soisalon-Soininen [14] proposed an approach, using which a smaller *NFA* can be generated than the Thompson's construction. The *NFA* generated using both the above approaches involves ε-transitions. If the *NFA* contains ε-transitions, the pattern matching

will delay due to the ε-transitions involved in the *NFA*. For fast pattern matching, *NFA* generated from *RE* should be ε-free and smaller in size. ε-free *NFA* can be constructed using different techniques named as Glushkov automata, follow automata and partial derivative automata [15-17]. Partial derivative automata generate a smaller ε-free *NFA* having at most $(m+1)$ states and m^2 transitions. Champarnaud and Ziadi [18] proposed an improved algorithm with $O(n^2)$ time complexity for the construction of partial derivative automata.

Estrade *et al*. [8] investigated the conversion of *PREs* to deterministic finite automata (*DFAs*). Gelade [7] worked on the succinctness of *REs* with shuffle, intersection and counting operator and proved that the double exponential size cannot be avoided for the conversion of *PREs* to *REs*. The proposed work of this paper is to generalize the partial derivative automata for the conversion of *PRE* to ε-free *NFA*.

The rest of the paper is organized as follows: section 2 contains necessary definitions and basic notations. In section 3, the concept of partial derivative automata is recalled. Section 4 is dedicated to the proposed approach for the conversion of *PREs* to ε-free *NFAs* by extending the partial derivative automata for the conversion of *REs* to ε-free NFAs. Section 5 contains the conclusions and future scope.

2. DEFINITIONS AND NOTATIONS

An alphabet (denoted by Σ) is a set of symbols and a language [3] is defined as a subset of Σ^*. A string x is a finite sequence of symbol taken from Σ. Length of a string x (denoted by $|x|$) is the total number of occurrence of different symbols in x. Empty string (having length *0*) is denoted by ε. Null language (denoted by ϕ) does not contain any string. A regular language can be represented by a *RE* or finite automaton. Finite automata can be deterministic or non-deterministic.

Definition 2.1 *RE* [3] over Σ can be defined by using following rules:

- *RE* for symbol $a \in \Sigma$ is represented by a.
- Null language (ϕ) and null string (ε) are represented by *REs* ϕ and ε respectively.
- Consider regular languages L_1 and L_2 are represented by *REs* r_1 and r_2 respectively, then *REs* r_1+r_2, r_1r_2 and r_1^* represent their union, concatenation and Kleene closure. Rule 3 can be applied recursively.

Definition 2.2 *DFA* [5] can be defined as a system $M=(S, R)$, where S is divided into two pair wise disjoint of states Q and an alphabet Σ. R contains a set of rules and each rule maps $Q \times \Sigma \to Q$. R is known as transition function. One state from Q is known as starting state (denoted by q_0). F is a set of final states and $F \subseteq Q$.

Definition 2.3 *NFA* [3] is similar as *DFA* except the state transition relation(R) is defined by $Q \times \Sigma \to 2^Q$. If a *NFA* does not contain any ε-transition, then the *NFA* is known as ε-free *NFA*.

Definition 2.4 Shuffling [3] of two strings x and y is a set which include all string z such that:

- Each position of z is assigned either to x or y but not to both at the same time.
- While reading string z from left to right, if we combine the symbols of the positions assigned to x in z, we obtain x.
- While reading string z from left to right, if we combine the symbols of the positions assigned to y in z, we obtain y.

Example 2.1 Given x=uv and y=ab, then x III y= {*uvab*, *uabv*, *uavb*, *abuv*, *auvb*, *aubv*}. If

L_1 and L_2 are two languages, then L_1 III L_2=$\{w \mid w=w_1$ III $w_2, \forall w_1\ L_1$ and $\forall w_2\ L_2\}$.

Definition 2.5 Parallel Finite Automata (*PFA*) [12] can be defined as 7-tuple (Q, Σ, q_0, F, γ, δ, N) where $Q \subseteq 2^N$ is a finite set of states; Σ is an alphabet; q_0 is the starting state; $F \subseteq N$ is the set of final nodes; γ is a node transition function, defined as $2^N \times (\Sigma \cup \lambda) \rightarrow 2^{2^{\wedge}n}$; δ is a transition function defined by $Q \times (\Sigma \cup \lambda) \rightarrow 2^Q$; N is a finite non-empty set of nodes. *PREs* can be represented by *PFAs*. λ-transition represents the joining of two languages with shuffle operator.

Definition 2.6 *First*(r) [1] can be defined as a set consisting of the position of the first symbols generated from *RE r*.

Definition 2.7 *Integer partition* [19] of a positive integer n is a non-increasing sequence of positive integers $p_1, p_2, \ldots, p_i$ such that $n=p_1+p_2+\ldots+p_i$. Each p_i is called a part of the partition.

Example 2.2 For $n=5$, following seven partitions exist:
i) 5 ii) 4+1 iii) 3+2 iv) 3+1+1 v) 2+2+1
vi) 2+1+1+1 vii) 1+1+1+1+1

Definition 2.8 Sub-regular expression denotes a part of the *PRE* which does not contain any shuffle operator and it contain only union, concatenation and Kleene closure operators.

Definition 2.9 *PRE r* is in unit-level of shuffling if shuffled sub-regular expressions of r will not get shuffled with other sub-regular expressions.

Definition 2.10 *PRE r* is in integration or multiple level of shuffling if shuffled sub-regular expressions of r will get shuffled with other sub-regular expressions.

Example 2.3 (a III b) c | (a III c) is an example of unit-level of shuffling. (a III b) III c is an example of multiple level of shuffling.

For the rest of the paper, m and k denote the total number of occurrence of symbols and level of shuffling in r. We use ordered integer partitions with a maximum value (n-1) and order of integers play a significant role.

3. PARTIAL DERIVATIVE AUTOMATA

Derivatives of *RE* were introduced by Brzozowski [20] in 1964, applying this algorithm *RE* can be converted into *DFA*. Later on, Antimirov [16] proposed the concept of partial derivatives for the conversion of *RE* to *NFA*.

Consider a language L over an alphabet $\{a, b\}$ denoted by *RE r*. Then, according to Antimirov [16] following are the properties of partial derivatives:

$$\partial_a(\varepsilon) = \phi \quad (1)$$
$$\partial_a(\phi) = \phi \quad (2)$$
$$\partial_a(a) = \varepsilon \quad (3)$$
$$\partial_a(b) = \phi \quad (4)$$
$$\partial_a(x \bigcup y) = \partial_a(x) \bigcup \partial_a(y) \quad (5)$$
$$\partial_a(x^*) = \partial_a(x)(x^*) \quad (6)$$

If $\varepsilon \in L(x)$ then

$$\partial_a(xy) = (\partial_a(x)y) \bigcup \partial_a(y) \quad (7)$$

If $\varepsilon \notin L(x)$ then

$$\partial_a(xy) = (\partial_a(x)y) \quad (8)$$

Starting state (q_0) and final state (F) of the *NFA* are defined by:

$q_0=r$ and $F=\{q \in Q \mid \varepsilon \in L(q)\}$.

Example 3.1 Consider the *RE* $r=(a+b)^*aa(a+b)^*$ over an alphabet $\Sigma=\{a, b\}$.
Using partial derivatives, *NFA* (illustrated in Figure 1) is computed as follow:

$r=(a+b)^*aa(a+b)^*=0$
$\partial_a((a+b)^*aa(a+b)^*)=$
$(\partial_a(a+b)^*)aa(a+b)^* \cup \partial_a(aa(a+b)^*)=$
$(a+b)^*aa(a+b)^* \cup (a(a+b)^*)= 0 \cup 1$
$\partial_b((a+b)^*aa(a+b)^*)=$

$(\partial_b(a+b)^*)aa(a+b)^* \cup \partial_b(aa(a+b)^*) =$
$(a+b)^*aa(a+b)^* \cup \phi = 0$
$\partial_a(a(a+b)^*) = \partial_a(a)(a+b)^* = (a+b)^* = 2$
$\partial_b(a(a+b)^*) = \phi$
$\partial_a((a+b)^*) = (a+b)^* = 2$
$\partial_b((a+b)^*) = (a+b)^* = 2$

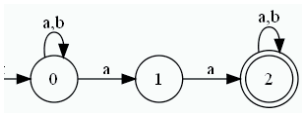

Figure 1. *NFA* for $r=(a+b)^*aa(a+b)^*$.

Proposition 3.1 Worst case *NFA* (*i.e. NFA* having $m+1$ states) occurs using partial derivatives, if m instances of symbols are connected by (m-1) concatenation operations and no other operators are used.

Proof. Consider *RE* $r=a_1a_2a_3...a_m$. Using partial derivative automata, $q_0=r$.

$\delta(q_0, a_1) = \partial_{a_1}(r) = a_2a_3...a_m = q_1$
$\delta(q_1, a_2) = \partial_{a_2}(q_1) = a_3...a_m = q_2$
$\delta(q_{m-1}, a_m) = \partial_{a_m}(q_{m-1}) = \varepsilon = q_m$

If m instance of symbols are connected by (m-1) concatenation operators and no other operators are used, then using partial derivatives ($m+1$) states are required. Next, we prove that the worst case cannot occur if r contains union or Kleene closure operator.

***RE r* contains Kleene closure:** Consider *RE* r is of the form $a_1a_2...a_{i-1}(a_i...a_j)^*a_{j+1}...a_m$.
Using partial derivatives,
$\delta(q_{i-1}, a_i) = \partial_{a_i}((a_i...a_j)^*a_{j+1}...a_m) =$
$(a_{i+1}\dots a_j)(a_ia_{i+1}\dots a_j)^*a_{j+1}...a_m = q_i$
$\delta(q_i, a_{i+1}) = \partial_{a_{i+1}}((a_{i+1}\dots a_j)(a_ia_{i+1}\dots a_j)^*a_{j+1}...a_m) =$
$(a_{i+2}\dots a_j)(a_ia_{i+1}\dots a_j)^*a_{j+1}...a_m = q_{i+1}$
$\delta(q_{j-1}, a_j) = \partial_{a_j}(a_j)(a_ia_{i+1}\dots a_j)^*a_{j+1}...a_m) = (a_ia_{i+1}\dots a_j)^*a_{j+1}...a_m = q_{i-1}$

Thus, each Kleene closure causes the number of states to be reduced by one. Hence, if r contains a Kleene closure, then it cannot have more than m states.

***RE r* contains union operator:** Consider *RE* r is of the form $a_1a_2...a_{i-1}(a_i...a_j|a_{j+1}...a_t)\ a_{t+1}...a_m$.
Using partial derivatives,
$\delta(q_{i-1}, a_i) = \partial_{a_i}((a_i...a_j|a_{j+1}...a_t)\ a_{t+1}...a_m) =$
$(a_{i+1}...a_j)\ a_{t+1}...a_m = q_i$
$\delta(q_i, a_{i+1}) = \partial_{a_{(i+1)}}((a_{i+1}...a_j)\ a_{t+1}...a_m) =$
$(a_{i+2}...a_j)\ a_{t+1}...a_m = q_{i+1}$
$\delta(q_{j-1}, a_j) = \partial_{a_j}((a_j)\ a_{t+1}...a_m) = a_{t+1}...a_m = q_j$
Similarly
$\delta(q_{i-1}, a_{j+1}) = \partial_{a_{(j+1)}}((a_i...a_j|a_{j+1}...a_t)\ a_{t+1}...a_m) =$
$((a_{j+2}...a_t)\ a_{t+1}...a_m) = q_{j+1}$
$\delta(q_{t-1}, a_t) = \partial_{a_t}((a_t)\ a_{t+1}...a_m) = a_{t+1}...a_m = q_j$

On reading a_j and a_t respectively from q_{j-1} and q_{t-1}, a new state q_j is generated. Hence the number of states will be decreased by one. Each union operation causes the number of states to be decrease by one. Hence, if *RE* contains union operator, it can have maximum m states using partial derivatives.

Proposition 3.2 *NFA* obtained using Thompson's construction will have the minimum number of states, if r contains only concatenation operator.

Proof. Using Thompson's construction [13], each symbol, Kleene closure and union operator can be represented by the addition of two new states, but each concatenation operator is carried out by merging of two states. If r only contains (m-1) concatenation operators, then the *NFA* constructed using Thompson's construction require ($m+1$) states. Best case of the *NFA* (*NFA* having the minimum number of states) using Thompson's construction occurs if r contains only concatenation operators.

4. PARTIAL DERIVATIVES FOR CONVERSION OF *PRE* TO *NFA*

Estrade *et al.* [8] investigated the conversion of *PREs* to *DFAs* using *PFAs* and *NFAs* as intermediate steps. *FLAT* tool kit [21] is available for the conversion of *PREs* to *DFAs*

through these intermediate steps as written in the sequence below:

$$PRE \rightarrow PFA \rightarrow NFA \rightarrow DFA \rightarrow DFA_{min}$$

Figure 3 represent *PFA* using Estrade *et al.* methodology for the *PRE* a^* III b^* and its corresponding *PFA*. λ-transition in Figure 3 represents the joining of two languages with shuffle operator.

Following are the properties of shuffle operator [8, 11]:

- Identity law: ε III $r = r$
- Associative law : $(r_1$ III $r_2)$ III $r_3 = r_1$ III $(r_2$ III $r_3)$
- Commutative law : $(r_1$ III $r_2) = (r_2$ III $r_1)$
- ε III $\varepsilon = \varepsilon$
- $(a$ III $b)^* = (a + b)^*$
- $(r_1$ III $r_2)$ III $(r_3$ III $r_4) = (r_1$ III r_2 III r_3 III $r_4)$

Lemma 4.1 For each symbol $a \in \Sigma$ and for regular terms x, y and z over Σ, following equations will hold:

a) $\partial_a(x$ III $y) = (\partial_a(x)$ III $y) \cup (x$ III $\partial_a(y))$

b) $\partial_a((x$ III $y)$ III $z) = (\partial_a(x)$ III y III $z) \cup (x$ III $(\partial_a(y))$ III $z) \cup (x$ III y III $(\partial_a(z))$

Proof. a) Let $r = x$ III y be a state at any moment of time. For each symbol, $a \in \Sigma$, we may or may not have input transitions from the state representing x III y.

If $a \in first(x)$, then $\delta(n, a) = (\partial_a(x)$ III $y)$

Similarly, if $a \in first(y)$ then $\delta(n, a) = (x$ III $\partial_a(y))$

Combining these using the property of shuffle, we get

$$\partial_a(x \text{ III } y) = (\partial_a(x) \text{ III } y) \cup (x \text{ III } \partial_a(y)).$$

b) $\partial_a((x$ III $y)$ III $z) =$
$((\partial_a(x$ III $y))$ III $z) \cup ((x$ III $y)$ III $\partial_a(z)) =$
$((\partial_a(x)$ III $y)$ III $z) \cup ((x$ III $\partial_a(y))$ III $z) \cup$
$((x$ III $y)$ III $\partial_a(z)) =$
$(\partial_a(x)$ III y III $z) \cup (x$ III $\partial_a(y)$ III $z) \cup$
$(x$ III y III $\partial_a(z))$.

Lemma 4.2 Consider a state $n_1 = (x$ III $y)$ z such that $\varepsilon \notin L(x$ III $y)$. On reading symbol a $\in \Sigma$, we have

a) If $\partial_a(x) = \phi$ and $\partial_a(y) \neq \phi$ then $\delta(n_1, a) = (x$ III $\partial_a(y))z = n_2$

b) If $\partial_a(x) = \partial_a(y) = \phi$ then $\delta(n_1, a) = \phi$

Proof. a) $a \notin first(x)$ implies $\partial_a(x) = \phi$ and a $\in$ $first(y)$ implies $\partial_a(y) \neq \phi$.

Given $\varepsilon \notin L(x$ III $y)$ and $\partial_a(x) = \phi$, implies that symbol a, cannot be generated from x and z. Using partial derivatives $\delta(n_1, a) = (x$ III $\partial_a(y))z = n_2$

b) $\partial_a(x) = \partial_a(y) = \phi$ means $a \notin first(x)$ and $a \notin first(y)$. It means symbol a, cannot be the first symbol generated from the *PRE* x III y. Using partial derivatives $\delta(n_1, a) = \phi$

Lemma 4.3 Consider a state $n_1 = (x$ III $y)z$ such that $\in L(x$ III $y)$. On reading a symbol $a \in \Sigma$, we have

a) If $\partial_a(x) = \phi$ and $\partial_a(y) \neq \phi$, then $\delta(n_1, a) = (x$ III $\partial_a(y))$ z $\cup$ $\partial_a(z) = n_2 \cup n_3$

b) If $\partial_a(x) = \partial_a(y) = \phi$, then $\delta(n_1, a) = \partial_a(z)$.

Proof. a) If $\varepsilon \in L(x$ III $y)$ then
$\partial_a(x$III$y)z = (\partial_a(x)$III$y)z \cup (x$ III $\partial_a(y))z \cup \partial_a(z)$
If $\partial_a(x) = \phi$, $\delta(n_1, a) = (x$III$\partial_a(y))z \cup \partial_a(z) = n_2 \cup n_3$
b) If $\varepsilon \in L(x$ III $y)$ then $\partial_a(x$ III $y)$ z $= (\partial_a(x)$ III $y)$ z $\cup$ $(x$ III $\partial_a(y))$ z $\cup$ $\partial_a(z)$
Now $\partial_a(x) = \partial_a(y) = \phi$, implies that $\delta(n_1, a) = \partial_a(z)$.

4.1 Numerical Examples

Example 4.1 Given *PRE* $r = ((a$ III $b)$ c III $ab)$ States of the *NFA* are represented by natural numbers and *NFA* is shown in Figure 4.

$r = ((a$ III $b)c$ III $ab) = 0$
$\partial_a(r) = (bc$ III $ab) \cup ((a$ III $b)c$ III $b) = 1 \cup 3$
$\partial_b(r) = ac$ III $ab = 2$
$\partial_c(r) = \phi$
$\partial_a(bc$ III $ab) = bc$ III $b = 4$
$\partial_b(bc$ III $ab) = c$ III $ab = 5$
$\partial_a(ac$ III $ab) = (c$ III $ab) \cup (ac$ III $b) = 5 \cup 6$

$\partial_a((a \text{ III } b)\ c \text{ III } b) = bc \text{ III } b = 4$
$\partial_b((a \text{III } b)\ c \text{III } b) = (ac \text{III } b) \cup ((a \text{III } b)\ c) = 6 \cup 7$
$\partial_b(bc \text{ III } b) = (c \text{ III } b) \cup bc = 9 \cup 8$
$\partial_a(c \text{ III } ab) = (c \text{ III } \partial_a(ab)) = c \text{ III } b = 9$
$\partial_c(c \text{ III } ab) = ab = 10$
$\partial_a(ac \text{ III } b) = c \text{ III } b = 9$
$\partial_b(ac \text{ III } b) = ac = 11$
$\partial_a((a \text{ III } b)\ c) = bc = 8$
$\partial_b((a \text{ III } b)\ c) = ac = 11$
$\partial_b(bc) = c = 12, \partial_b(c \text{ III } b) = c = 12$
$\partial_c(c \text{ III } b) = b = 13, \partial_a(ab) = b = 13$
$\partial_a(ac) = c = 12, \partial_b(b) = \varepsilon = 14$
$\partial_c(c) = \varepsilon = 14$

Example 4.2 Given *PRE* $r = (a \text{ III } b)^* \text{III } c^*$
Starting state $0 = (a \text{ III } b)^* \text{ III } c^*$
$\partial_a((a \text{ III } b)^* \text{III } c^*) = (b(a \text{ III } b)^*) \text{III } c^* = 1$
$\partial_b((a \text{ III } b)^* \text{ III } c^*) = (a(a \text{ III } b)^*) \text{III } c^* = 2$
$\partial_c((a \text{ III } b)^* \text{ III } c^*) = (a \text{ III } b)^* \text{ III } c^* = 0$
$\partial_a((b(a \text{ III } b)^*) \text{ III } c^*) = \phi$
$\partial_b((b(a \text{ III } b)^*) \text{ III } c^*) = (a \text{ III } b)^* \text{ III } c^* = 0$
$\partial_c((b(a \text{ III } b)^*) \text{ III } c^*) = ((b(a \text{ III } b)^*) \text{ III } c^*) = 1$
$\partial_a((a(a \text{ III } b)^*) \text{ III } c^*) = (a \text{ III } b)^* \text{III } c^* = 0$
$\partial_c((a(a \text{ III } b)^*) \text{ III } c^*) = ((a(a \text{ III } b)^*) \text{ III } c^*) = 2$

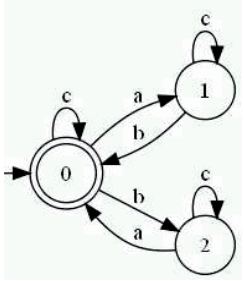

Figure 2. *NFA* for $r = (a \text{ III } b)^* \text{ III } (c)^*$.

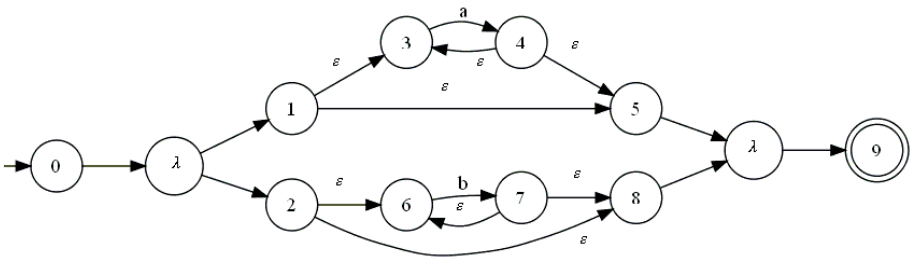

Figure 3. *PFA* for $r = a^* \text{ III } b^*$.

Table 1 and Table 2 show the precedence and associativity of operators.

Table 1. Operator precedence while one operator distributed over other operators.

S.No.	Description	Expression type	Example	Preference
1.	Shuffle operator distributed over other operators	$r_1 \text{ III } (r_2 \mid r_3)$	i) $a \text{ III } (b \mid c)$	Shuffle
		$(r_1)^* \text{ III } (r_2)^*$	i) $a \text{ III } b^*$ ii) $(a \text{ III } b)^* \text{ III } c^*$ iii) $a^* \text{ III } b^*$	
		$(r_1 r_2) \text{III} (r_3 r_4)$	i) $(ab) \text{ III } (cd)$	
2.	Union operator distributed over other operators	$r_1 \mid (r_2 \text{ III } r_3)$	i) $c \mid (a \text{ III } b)$	Union
		$r_1 \mid r_2$	i) $(ab) \mid (cd)$	
		$(r_1)^* \mid (r_2)^*$	i) $a^* \mid c^*$	
3.	Kleene closure operator over other operators	$(r_1 \text{ III } r_2)^*$	i) $(a \text{ III } b)^*$	Kleene closure
		$(r_1 r_2)^*$	i) $(ab)^*$	
		$(r_1 \mid r_2)^*$	i) $(ab \mid bc)^*$	
4.	Concatenation	$(r_1 \mid r_2)c$	i) $(a \mid b)\ c$	Union
		$(r_1 \text{ III } r_2)c$	i) $(a \text{ III } b)c$	Shuffle
		$(r_1 \text{ III } r_2)^*\ r_3$	i) $(a \text{ III } b)^* c$	Kleene closure

Table 2. Operator's precedence and associativity in *PRE*.

S.No.	Operator Name	Associativity	Example	Preference
1.	Shuffle	Right to Left	a III $b \mid c$	III
2.	Union	Right to Left	$a^* \mid bc$	$\mid$
3.	Kleene closure	Right to Left	a^*b	*
4.	Concatenation	Left to Right	abc	First concatenation

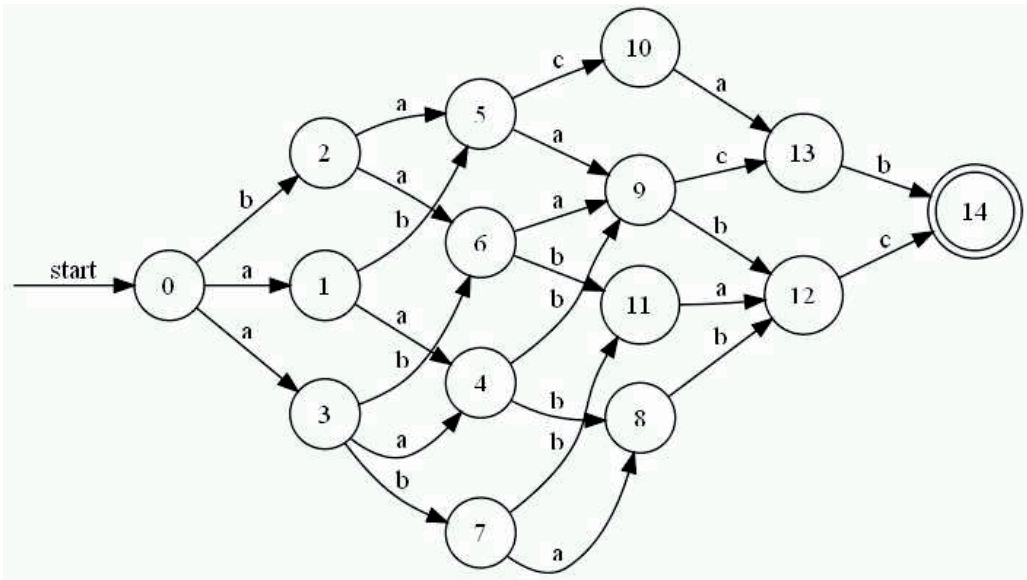

Figure 4. *NFA* for $r = ((a \text{ III } b)\ c \text{ III } ab)$ using the proposed approach.

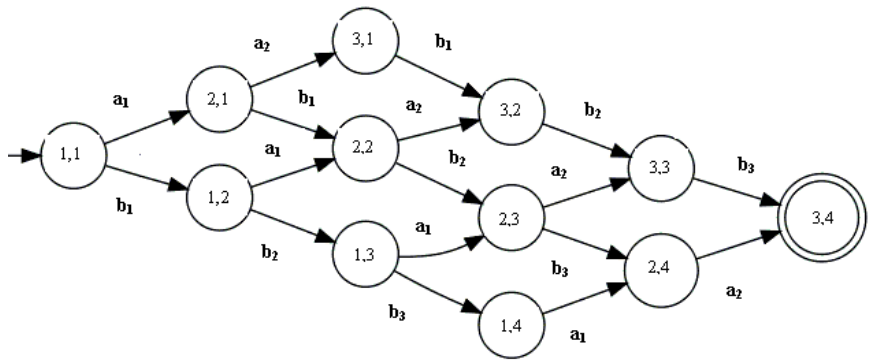

Figure 5. *NFA* for $r=(a_1a_2)\text{III}(b_1b_2b_3)$.

Lemma 4.4 If *PRE r* consists of unit-level of shuffling. Then, using partial derivatives, the number of states obtained in the *NFA* is always less than $(m+1)\ (m+2)/2$.

Proof. Using partial derivatives, worst case can occur, if *PRE* $r = (a_1a_2\ldots a_i)\text{III}\ (b_1b_2\ldots b_j)$ and $m = (i+j)$.Starting state $q_0 = (a_1a_2\ldots a_i)$ III $(b_1b_2\ldots b_j)$.We can represent the starting state by $(1,1)$ indicating that we are going to read a_1 and b_1 from state q_0.

$$\delta(q_0, a_1)=\partial_{a_1}(r)=(a_2\ldots a_i)\text{III}(b_1b_2\ldots b_j)=(2, 1)$$
$$\delta(q_0, b_1)=\partial_{b_1}(r)=(a_1a_2\ldots a_i)\text{III}(b_2\ldots b_j)=(1, 2)$$

Two new states are generated on reading of the first symbol. The number of new states generated is equal to the ordered integer partition of three, such that the maximum number is two, and the number of parts in the partition is two. $T_{i,j}$ denote the number of ordered integer partitions of j with the maximum number $(j\text{-}1)$ and each partition consists of exactly i parts. If sum of k numbers is n, then the range for picking k number is $[(n\text{-}k\text{+}1)\ldots 1]$.

If $k=2$ and $sum=3$, then the range of each number is $[2, 1]$. On reading of the second symbol, following additional states are generated:

$\delta((2,1),a_2)=\partial_{a_2}(2,1)=(a_3\ldots a_i)\text{III}(b_1b_2\ldots b_j)=(3,1)$

$\delta((2,1),b_1)=\partial_{b_1}(2,1)=(a_2a_3\ldots a_i)\text{III}(b_2\ldots b_j)=(2,2)$

$\delta((1,2),a_1)=\partial_{a_1}(1,2)=(a_2a_3\ldots a_i)\text{III}(b_2\ldots b_j)=(2,2)$

$\delta((1,2),b_2)=\partial_{b_2}(1,2)=(a_1a_2a_3\ldots a_i)\text{III}(b_3\ldots b_j)=(1,3)$

$T_{2,4}=3$

Repeating the above process, $T_{2,m+2}=(m+1)$

Total number of states in the *NFA* is

$$\sum_{i=2}^{m+2}(T_{2,i})=(m+1)(m+2)/2=O(m^2)$$

In actual, number of states, obtained is always less than $(m+1)(m+2)/2$. After reading m^{th} symbol, we consider $(m+2)$ new state, but in actual there is only one final state. Suppose $i<j$ and after reading i^{th} symbol, $T_{2,(i+2)}=(i+1)$ new states are generated. We have no other options, but to read b_1 on state represented by $(i+1, 1)$. Hence $(i+1)$ new states are generated for reading i^{th} to j^{th} symbol as shown in Figure 5. Number of new states generated on reading of $(j+1)^{th}$ and onwards symbols can be represented by $T_{2,(j+2+s)}=(i+1)\text{-}s$.

The number of states generated in the worst case of a *NFA* is equal to $ij+j+i+1$. Clearly $(ij+j+i+1)<(m+1)(m+2)/2$ where $m=(i+j)$. Hence the number of states required for the *NFA* is always less than $(m+1)(m+2)/2$.

For finding the number of states in multi-level shuffling, we consider the number of states in unit-level is equal to $(m+1)\ (m+2)/2$.

Lemma 4.5 If *PRE r* consists of *2*-level of shuffling. Using partial derivatives, the number of states obtained in the *NFA* is $O(m^3)$.

Proof. Consider *PRE r* is of the form of $(a_1\ldots a_i)\ \text{III}\ (b_1b_2\ldots b_j)\ \text{III}\ (c_1c_2\ldots c_k)$ and $m=(i+j+k)$. Starting state q_0 is equal to r and represented by $(1, 1, 1)$.

$\delta(q_0, a_1)=\partial_{a_1}(r)=(a_2\ldots a_i)\text{III}(b_1b_2\ldots b_j)\ \text{III}(c_1c_2\ldots c_k)=(2, 1, 1)$

$\delta(q_0, b_1)=\partial_{b_1}(r)=(a_1a_2\ldots a_i)\text{III}(b_2\ldots b_j)\ \text{III}(c_1c_2\ldots c_k)=(1, 2, 1)$

$\delta(q_0, c_1)=\partial_{c_1}(r)=(a_1a_2\ldots a_i)\text{III}(b_1b_2\ldots b_j)\ \text{III}(c_2\ldots c_k)=(1, 1, 2)$

Clearly range of picking three numbers is $[4\text{-}3+1, \ldots, 1]$

$T_{3,4}=3=T_{2,2}+T_{2,3}$

On reading second symbol, the number of new state generated is equal to

$T_{3,5}=T_{2,4}+T_{2,3}+T_{2,2}$

Repeating the above process, if the sum of three numbers is n then

$T_{3,n}=T_{2,n-1}+T_{2,n-2}+\ldots+T_{2,2}=(n-2)(n-1)/2$

Number of states in the *NFA* is $\sum_{i=3}^{m+3}(T_{3,i})=m^3/6+m^2+11m/6+1=O(m^3)$

Theorem 4.1 If *PRE r* consists of *k*-level of shuffling. Using partial derivatives, the number of states obtained in the *NFA* is $O(m^{k+1})$.

Proof. If *r* consists of *k*-level of shuffling, the starting state will be represented by *k*-parts whose sum is *k*. If sum of *k* numbers is *n*, then range of the *k* numbers is $[n\text{-}k+1, \ldots, 1]$. For $n>4$, $T_{4,n}=T_{3,n-1}+T_{3,n-2}+\ldots+T_{3,3}$ and the number of new states after reading each symbol is of the form of n^3.

If $n=4$

$T_{4,4}=1$

Similarly
For $n>k$, $T_{k,n} = T_{k-1,n-1} + T_{k-1,n-2} + \ldots + T_{k-1,k-1}$ and is a polynomial of the form of n^k.
If $n=k$, $T_{k,n}=1$
By using the Bernoulli's formula [22], for summing the series of natural number, we obtain a polynomial of degree $(k+1)$. Hence the number of states obtained in the *NFA* is $O(m^{k+1})$.

5. Conclusions and Future Scope

In the proposed work, the conversion of *PREs* to *NFAs* is carried out using partial derivatives. The number of states generated using this approach in the worst case is $O(m^{k+1})$, which is a significant improvement as compared to other existing approaches till date. Further, this work can be extended for finding the exact number of states in multi-level shuffling.

Acknowledgements

The authors are grateful to the anonymous Referees, Editor and Editorial staff for a careful checking of the details and their valuable comments and suggestion significantly improved the paper.

References

[1] Aho A.V., Sethi R. and Ullman J. D., *Compilers: Principles, Techniques and Tools*, 19th Edn., Pearson Education, 2005.

[2] Clark J. and Murata M., RELAX NG Specifications, Available at: http://www.oasis-open.org/committes/relax-ng/spec-20011203.html, Accessed 2001.

[3] Hopcroft J. E., Motwani R. and Ullman J. D., *Introduction to Automata Theory, Languages and Computation*, 12th Edn., Pearson Education, 2005.

[4] Wall L., Christiansen T. and Orwant J., Programming Perl, O'Reilly Media, 3rd Edn., 2000.

[5] Meduna A., *Elements of Compiler Designs*, 1st Edn., Auerbach Publications, 2007.

[6] Stotts P. D. and Pugh W., Parallel finite automata for modeling concurrent software systems, *J. Systems Softw.*, 1994; **27(1)**: 27-43.

[7] Gelade W., Succinctness of regular expressions with interleaving, intersection and counting, *Theoretical Comput. Sci.*, 2010; **411(31-33)**: 2987-2998.

[8] Estrade B. D., Perkins A. L. and Harris J. M., Explicitly Parallel Regular Expressions, *Proceedings of the 1st International Multi-Symposiums on Computer and Computational Sciences (IMSCCS06)*, Hangzhou, China, IEEE, 2006; 402-409.

[9] Baeten J.C.M. and Weijland W.P., *Process Algebra*, *Cambridge Tract in Theoretical Computer Science*, Cambridge University Press, Cambridge, 1990.

[10] Iwama K., Unique Decomposability of Shuffled Strings: A Formal Treatment of Asynchronous Time-multiplexed Communication, *Proceedings of the 15th Annual ACM Symposium on Theory of Computation*, Boston, USA, 1983; 374-378.

[11] Estrade B. D., *An Investigation of Equivalent Serialized Forms of Parallel Finite Automata*, MS Thesis, The University of Southern Mississippi, USA, 2005.

[12] Garg V. K., Modeling of Distributed Systems by Concurrent Regular Expressions, *Proceedings of the 2nd International Conference on Formal Description techniques for Distributed Systems and Communication Protocols*, 1989.

[13] Thompson K., Regular expression search algorithm, *Communications of the ACM*, 1968; **11(6)**: 419-422.

[14] Sippu S. and Soisalon-Soininen E., *Parsing Theory: vol. 1, Languages and Parsing*, *EATCS*

Monographs on Theoretical Computer Science, Springer-Verlag, New York, 1988.

[15] Ilie L. and Yu S., Follow automata, *Inform. Comput.*, 2003; **186(1)**: 146-162.

[16] Antimirov V., Partial derivatives of regular expressions and finite automaton constructions, *Theoretical Comput. Sci.*, 1996; **155**: 291-319.

[17] Glushkov V. M., The abstract theory of automata, *Russian Math. Surveys*, 1961; **16(5)**: 1-53.

[18] Champarnaud J.M. and Ziadi D., Computing the Equation Automation of Regular Expression in $O(s^2)$ Space and Time, *Proceedings of Combinatorial Pattern Matching 2001*, Springer-Verlag, 2001; 2089: 157-168.

[19] http://www.math.upenn.edu/~wilf/PIMS/PIMSLectures.pdf

[20] Brzozowski J.A., Derivatives of regular expressions, *J. Assoc. Comput. Mach.*, 1964; **11(4)**: 481-494.

[21] http://search.cpan.org/~estrabd/FLAT-0.9.1/lib/FLAT.pm

[22] http://www.math.hawaii.edu/~pavel/bernoulli.pdf