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 correspondence; 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 $O(2^n)$. 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 $\varepsilon$-free non-deterministic finite automata. The number of states of the non-deterministic finite automaton is reduced from exponential ($O(2^n)$) to polynomial ($O(m^{2n})$), using this novel approach.

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

1. Introduction

Regular expressions ($RE$s) 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]. Extended $RE$s 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 $\varepsilon$-transitions. If the $NFA$ contains $\varepsilon$-transitions, the pattern matching...
will delay due to the \( \epsilon \)-transitions involved in the \( NFA \). For fast pattern matching, \( NFA \) generated from \( RE \) should be \( \epsilon \)-free and smaller in size. \( \epsilon \)-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 \( \epsilon \)-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 \( \epsilon \)-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 \( \epsilon \)-free \( NFAs \) by extending the partial derivative automata for the conversion of \( REs \) to \( \epsilon \)-free \( NFAs \). Section 5 contains the conclusions and future scope.

2. DEFINITIONS AND NOTATIONS

An alphabet (denoted by \( \Sigma \)) is a set of symbols and a language [3] is defined as a subset of \( \Sigma^* \). A string \( x \) is a finite sequence of symbol taken from \( \Sigma \). 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 \( \epsilon \). Null language (denoted by \( \phi \)) 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 \( \Sigma \) can be defined by using following rules:

- \( RE \) for symbol \( a \in \Sigma \) is represented by \( a \).
- Null language (\( \phi \)) and null string (\( \epsilon \)) are represented by \( REs \) \( \phi \) and \( \epsilon \) respectively.
- Consider regular languages \( L_1 \) and \( L_2 \) are represented by \( REs \) \( r_1 \) and \( r_2 \) respectively, then \( REs r_1 + r_2 \), \( r_1 r_2 \) 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 \( \Sigma \). \( R \) contains a set of rules and each rule maps \( Q \times \Sigma \rightarrow Q \). \( R \) 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 \( Q \times \Sigma \rightarrow 2^Q \). If a \( NFA \) does not contain any \( \epsilon \)-transition, then the \( NFA \) is known as \( \epsilon \)-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 \( x \text{ III } y = \{uvab, uabr, uavb, abuv, avub, anvb\} \). If
Definition 2.5 Parallel Finite Automata (PRE) [12] can be defined as 7-tuple \((Q, \Sigma, q_0, F, \gamma, \delta, N)\) where \(Q \subseteq 2^N\) is a finite set of states; \(\Sigma\) is an alphabet; \(q_0\) is the starting state; \(F \subseteq N\) is the set of final nodes; \(\gamma\) is a node transition function, defined as \(2^\gamma \times (\Sigma \cup \lambda) \rightarrow 2^\gamma\); \(\delta\) 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 \(\text{First}(r)\) [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.

Example 2.3 \((a \text{ III } b) c\) is an example of unit-level of shuffling, \((a \text{ III } c)\) 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:

\begin{align*}
\partial_a(\varepsilon) &= \emptyset \\
\partial_a(\phi) &= \emptyset \\
\partial_a(a) &= \varepsilon \\
\partial_a(b) &= \emptyset \\
\partial_a(x \cup y) &= \partial_a(x) \cup \partial_a(y) \\
\partial_a(x^*) &= \partial_a(x)(x^*)
\end{align*}

If \(\varepsilon \in L(x)\) then
\(\partial_a(xy) = (\partial_a(x)y) \cup \partial_a(y)\) (7)

If \(\varepsilon \notin L(x)\) then
\(\partial_a(xy) = (\partial_a(x)y)\) (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 \(RE\) \(r=(a+b)^*ad(a+b)^*\) over an alphabet \(\Sigma = \{a, b\}\).

Using partial derivatives, \(NFA\) (illustrated in Figure 1) is computed as follow:

\begin{align*}
&\partial_a((a+b)^*ad(a+b)^*) = \\
&\partial_a(ad(a+b)^*) = \\
&\partial_a((a+b)^*ad(a+b)^*) = \\
&\partial_a(ad(a+b)^*) = \\
&\partial_a(ad(a+b)^*) =
\end{align*}
\((\partial_{b}(a+b)^*)aa(a+b)^* \cup \partial_{a}(aa(a+b)^*) = (a+b)^*aa(a+b)^* \cup \phi = 0\)

\(\partial_{a}(a(a+b)^*) = \partial_{a}(a(a+b)^*) = (a+b)^* = 2\)

\(\partial_{b}(a(a+b)^*) = \partial_{b}(a(a+b)^*) = (a+b)^* = 2\)

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, \(q_0=r\).

\(\delta(q_0, a) = \partial_{a}(r) = a_1a_2a_3...a_m = q_1\)

\(\delta(q_1, a) = \partial_{a}(q_1) = a_2a_3...a_m = q_2\)

\(\delta(q_2, a) = \partial_{a}(q_2) = \partial_{a}(a_3...a_m) = \epsilon = q_m\)

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 \(a_1a_2...a_k(a_1a_2...a_l)^*a_{k+1}...a_m\).

Using partial derivatives,

\(\delta(q_j, a) = \partial_{a}(q_j) = (a_1a_2...a_k)^*a_{j+1}...a_m = q_j\)

\(\delta(q_j, a) = \partial_{a}(q_j) = (a_1a_2...a_k)^*a_{j+1}...a_m = q_{j+1}\)

\(\delta(q_j, a) = \partial_{a}(q_j) = (a_1a_2...a_k)^*a_{j+1}...a_m = q_{j+1}\)

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 \(a_1a_2...a_k(a_1a_2...a_l) a_{k+1}...a_m\).

Using partial derivatives,

\(\delta(q_j, a) = \partial_{a}(q_j) = (a_1a_2...a_k) a_{j+1}...a_m = q_j\)

\(\delta(q_j, a) = \partial_{a}(q_j) = (a_1a_2...a_k) a_{j+1}...a_m = q_{j+1}\)

\(\delta(q_j, a) = \partial_{a}(q_j) = (a_1a_2...a_k) a_{j+1}...a_m = q_{j+1}\)

Similarly

\(\delta(q_j, a) = \partial_{a}(q_j) = (a_1a_2...a_k) a_{j+1}...a_m = q_{j+1}\)

On reading \(a_j\) and \(a_i\) respectively from \(q_{j+1}\) and \(q_{j+1}\), 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 (NEA 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:

\[ \text{PRE} \rightarrow \text{PFA} \rightarrow \text{NEA} \rightarrow \text{DFA} \rightarrow \text{DFA}_{\min} \]

Figure 3 represent PFA using Estrade et al. methodology for the \text{PRE} \ a I I I b \ and its corresponding PFA. \( \lambda \)-transition in Figure 3 represents the joining of two languages with shuffle operator.

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

- Identity law: \( \varepsilon \ I I I r = r \)
- Associative law: \((r_1 I I I r_2) I I I r_3 = r_1 I I I (r_2 I I I r_3)\)
- Commutative law: \((r_1 I I I r_2) = (r_2 I I I r_1)\)
- \( \varepsilon \ I I I \varepsilon = \varepsilon \)
- \((a I I I b)^n = (a^n b^n)\)
- \((r_1 I I I r_2) I I I (r_3 I I I r_4) = (r_1 I I I r_3) I I I (r_2 I I I r_4)\)

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

a) \( \partial_r (x I I I y) = (\partial_r (x) I I I y) U (x I I I \partial_r (y)) \)
b) \( \partial_r ((x I I I y) I I I z) = (\partial_r (x) I I I y I I I z) U (x I I I (\partial_r (y)) I I I z) \) U \((x I I I (\partial_r (y)) I I I \partial_r (z))\)  

**Proof.**  
a) Let \( r = x I I I y \) 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 \( x I I I y \).

If \( a \in \text{first}(x) \), then \( \partial_r (n, a) = (\partial_r (x) I I I y) \)

Similarly, if \( a \in \text{first}(y) \) then \( \partial_r (n, a) = (x I I I \partial_r (y)) \)

Combining these using the property of shuffle, we get

\[ \partial_r (x I I I y) = (\partial_r (x) I I I y) \cup (x I I I \partial_r (y)) \]

b) \( \partial_r ((x I I I y) I I I z) = (\partial_r (x) I I I y) I I I z) \cup ((x I I I y) I I I \partial_r (z)) = (\partial_r (x) I I I y) I I I \partial_r (z) \cup ((x I I I y) I I I \partial_r (z)) \)

**Lemma 4.2** Consider a state \( n = (x I I I y) z \) such that \( \varepsilon \not\in L(x I I I y) z \). On reading symbol \( a \in \Sigma \), we have

a) If \( \partial_r (x) = \phi \) and \( \partial_r (y) \neq \phi \) then \( \partial(n, a) = (x I I I \partial_r (y) z) \)

b) If \( \partial_r (x) = \partial_r (y) = \phi \) then \( \partial(n, a) = \phi \)

**Proof.**  
a) \( a \not\in \text{first}(x) \) implies \( \partial_r (x) = \phi \) and \( a \in \text{first}(y) \) implies \( \partial_r (y) \neq \phi \).  

Given \( \varepsilon \not\in L(x I I I y) \) and \( \partial_r (x) = \phi \), implies that symbol \( a \), cannot be generated from \( x \) and \( z \). Using partial derivatives \( \partial(n, a) = (x I I I \partial_r (y) z) = n_2 \)

b) \( \partial_r (x) = \partial_r (y) = \phi \) means \( a \in \text{first}(x) \) and \( a \in \text{first}(y) \). It means symbol \( a \), cannot be the first symbol generated from the \( \text{PRE} \ x I I I y \).

Using partial derivatives \( \partial(n, a) = \phi \)

**Lemma 4.3** Consider a state \( n = (x I I I y) z \) such that \( \varepsilon \in L(x I I I y) z \). On reading a symbol \( a \in \Sigma \), we have

a) If \( \partial_r (x) = \phi \) and \( \partial_r (y) \neq \phi \), then \( \partial(n, a) = (x I I I \partial_r (y) z) \cup \partial_r (x) = n_1 \cup n_2 \)

b) If \( \partial_r (x) = \partial_r (y) = \phi \), then \( \partial(n, a) = \partial_r (z) \).

**Proof.**  
a) If \( \varepsilon \in L(x I I I y) z \) then \( \partial_r (x) I I I y) z = (\partial_r (x) I I I y) z \cup (x I I I \partial_r (y) z) \cup \partial_r (x) \)

If \( \partial_r (x) = \phi \), then \( \partial(n, a) = (x I I I \partial_r (y) z) \cup \partial_r (x) \).

b) If \( \varepsilon \in L(x I I I y) z \) then \( \partial_r (x) I I I y) z = (\partial_r (x) I I I y) z \cup (x I I I \partial_r (y) z) \cup \partial_r (x) \)

Now \( \partial_r (x) = \partial_r (y) = \phi \), implies that \( \partial(n, a) = \partial_r (z) \).

4.1 Numerical Examples

**Example 4.1** Given \( \text{PRE} \ r = ((a I I I b) \varepsilon I I I ab) \)

States of the NEA are represented by natural numbers and NEA is shown in Figure 4.

\[ r = ((a I I I b) \varepsilon I I I ab) = 0 \]

\( \partial_r (r) = (bc I I I ab) \cup ((a I I I b) I I I bc I I I ab) = 1 \cup 3 \)

\( \partial_r (r) = ac I I I ab = 2 \)

\( \partial_r (r) = \phi \)

\( \partial_r (bc I I I ab) = bc I I I b = 4 \)

\( \partial_r (bc I I I ab) = (a I I I b) \cup (ac I I I b) = 5 \cup 6 \)
\[ \partial((a \text{ III } b) \text{ III } c) = bc \text{ III } b = 4 \]
\[ \partial((a \text{ III } b) \text{ III } b) = (a \text{ III } b) \cup ((a \text{ III } b) \text{ III } b) = 6 \cup 7 \]
\[ \partial((a \text{ III } b) b) = (c \text{ III } b) b = 9 \cup 8 \]
\[ \partial(c \text{ III } ab) = (c \text{ III } b) c = 8 \text{ III } b = 9 \]
\[ \partial(ac \text{ III } b) = ab = 10 \]
\[ \partial(ac \text{ III } b) = c \text{ III } b = 9 \]
\[ \partial(ac \text{ III } b) = ac = 11 \]
\[ \partial(ab) = ac = 11 \]
\[ \partial(bc) = \epsilon = 12, \partial(c \text{ III } b) = \epsilon = 12 \]
\[ \partial(c \text{ III } b) = b = 13, \partial(ab) = b = 13 \]
\[ \partial(ab) = \epsilon = 12, \partial(b) = \epsilon = 14 \]
\[ \partial(c) = \epsilon = 14 \]

**Example 4.2**

Given \( PRE r = (a \text{ III } b) \text{ III } c \)

Starting state \( 0 = (a \text{ III } b) \) \text{ III } \( \epsilon \)

\[ \partial((a \text{ III } b) \text{ III } c) = (b(a \text{ III } b)^\ast) \text{ III } \epsilon = 1 \]

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

<table>
<thead>
<tr>
<th>S.No.</th>
<th>Description</th>
<th>Expression type</th>
<th>Example</th>
<th>Preference</th>
</tr>
</thead>
<tbody>
<tr>
<td>1.</td>
<td>Shuffle operator distributed over other operators</td>
<td>( r_1 \text{ III } (r_2</td>
<td>r_3) )</td>
<td>( i ) a \text{ III } (b</td>
</tr>
<tr>
<td></td>
<td></td>
<td>( (r_1)^\ast \text{ III } (r_2)^\ast )</td>
<td>( i ) a \text{ III } b^\ast \</td>
<td>\</td>
</tr>
<tr>
<td></td>
<td></td>
<td>( (r_1</td>
<td>r_2) \text{ III } (r_3) )</td>
<td>( i ) a \text{ III } b \</td>
</tr>
<tr>
<td>2.</td>
<td>Union operator distributed over other operators</td>
<td>( r_1</td>
<td>(r_2 \text{ III } r_3) )</td>
<td>( i ) \epsilon</td>
</tr>
<tr>
<td></td>
<td></td>
<td>( (r_1</td>
<td>r_2)^\ast \text{ III } (r_3)^\ast )</td>
<td>( i ) \text{ III } \epsilon \</td>
</tr>
<tr>
<td>3.</td>
<td>Kleene closure operator over other operators</td>
<td>( (r_1</td>
<td>r_2)^\ast \text{ III } (r_3)^\ast )</td>
<td>( i ) \text{ III } b \</td>
</tr>
<tr>
<td></td>
<td></td>
<td>( (r_1</td>
<td>r_2)^\ast \text{ III } (r_3)^\ast )</td>
<td>( i ) \text{ III } (b</td>
</tr>
</tbody>
</table>

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

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

Table 1 and Table 2 show the precedence and associativity of operators.
Table 2. Operator’s precedence and associativity in PRE.

<table>
<thead>
<tr>
<th>S.No.</th>
<th>Operator Name</th>
<th>Associativity</th>
<th>Example</th>
<th>Preference</th>
</tr>
</thead>
<tbody>
<tr>
<td>1.</td>
<td>Shuffle</td>
<td>Right to Left</td>
<td>a III b</td>
<td>III</td>
</tr>
<tr>
<td>2.</td>
<td>Union</td>
<td>Right to Left</td>
<td>a | bc</td>
<td>|</td>
</tr>
<tr>
<td>3.</td>
<td>Kleene closure</td>
<td>Right to Left</td>
<td>a^b</td>
<td>*</td>
</tr>
<tr>
<td>4.</td>
<td>Concatenation</td>
<td>Left to Right</td>
<td>abc</td>
<td>First concatenation</td>
</tr>
</tbody>
</table>

Figure 4. NFA for \( r = ((a III b) c III ab) \) using the proposed approach.

Figure 5. NFA for \( r = (a_1 a_2) III (b_1 b_2 b_3) \).

**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 \( PRE \ r = (a_1 a_2 \ldots a_i) III (b_1 b_2 \ldots b_j) \) and \( m = (i+j) \). Starting state \( q_0 = (a_1 a_2 \ldots a_i) III (b_1 b_2 \ldots b_j) \). We can represent the starting state by \((1,1) \) indicating that we are going to read \( a_i \) and \( b_j \) from state \( q_0 \).

\[
\delta(q_0, a_i) = \delta_1(i) = (a_1 a_2 \ldots a_i) III (b_1 b_2 \ldots b_j) = (2, 1)
\]

\[
\delta(q_0, b_j) = \delta_0(j) = (a_1 a_2 \ldots a_i) III (b_1 b_2 \ldots b_j) = (1, 2)
\]
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 \((j-1)\) and each partition consists of exactly \( i \) parts. If sum of \( k \) numbers is \( n \), then the range for picking \( k \) number is \([u-k+1] \ldots [l] \).

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

\[
\delta (2, i) = \partial \delta (2, i) = (a_{1} \ldots a_{2} \ldots b) = (3, 1)
\]

On reading second symbol, the number of new states after reading each symbol, following additional states can be generated:

\[
\delta (3, i) = \partial \delta (3, i) = (a_{1} \ldots a_{2} \ldots c) = (2, 2)
\]

The number of states generated in the NFA is equal to \((m+1)(m+2)/2 \).

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 \( \text{PRE} \ r \) consists of \( 2 \)-level of shuffling. Using partial derivatives, the number of states obtained in the NFA is \( O(m^{3}) \).

**Proof.** Consider \( \text{PRE} \ r \) is of the form of \((a_{1} \ldots a) \) \((b_{1} \ldots b) \) \((c_{1} \ldots c) \) and \( m=(i+j+k) \).

Starting state \( q_{0} \) is equal to \( r \) and represented by \((1, 1, 1) \).

\[
\delta (q_{0}, a) = \partial \delta (q_{0}, a) = (a_{1} \ldots a \ldots b) = (2, 2)
\]

\[
\delta (q_{0}, b) = \partial \delta (q_{0}, b) = (a_{1} \ldots a \ldots c) = (2, 2)
\]

Clearly range of picking three numbers is \([4, 3+1, \ldots, 1] \)

\[
T_{3, 4} = 3 = T_{2, 2} + T_{2, 3}
\]

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 \( \sum_{i=3}^{n} T_{3, i} = m^{3}/6 + m^{2} + 11m/6 + 1 = O(m^{3}) \)

**Theorem 4.1** If \( \text{PRE} \ r \) 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 \((i+k, l) \).

Hence \((i+k)\) new states are generated for reading \( i^{k} \) to \( j^{k} \) symbol as shown in Figure 5. Number of new states generated on reading of \((j+1)^{k}\) and onwards symbols can be represented by \( T_{3, (j+2)^{k}} = (i+1)-s \).

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

If \( n=4 \)

\[
T_{4, 4} = 1
\]
Similarly
For \( n > k \), \( T_{k,n} = T_{k,1,\ldots,1} + T_{k,1,\ldots,2} + \ldots + T_{k,1,k-1} \) and is a polynomial of the form of \( n^k \).
If \( n = k \), \( T_{k,k} = 1 \)
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


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