On The Space Complexity of Partial Derivatives of Regular Expressions with Shuffle

Davide Ancona[1,*], Angelo Ferrando[2]

[1]University of Genova, Italy
[2]University of Modena and Reggio Emilia, Italy

Abstract

Partial derivatives of regular expressions, introduced by Antimirov, define an elegant algorithm for generating equivalent non-deterministic finite automata (NFA) with a limited number of states. Here we focus on runtime verification (RV) of simple properties expressible with regular expressions. In this case, words are finite traces of monitorable events forming the language's alphabet, and the generated NFA may have an intractable number of states. This typically occurs when sub-traces of mutually independent events are allowed to interleave. To address this issue, regular expressions used for RV are extended with the shuffle operator to make specifications more compact and easier to read. Exploiting partial derivatives enables a rewriting-based approach to RV, where only one derivative is stored at each step, avoiding the construction of an intractably large automaton. This raises the question of the space complexity of the largest generated partial derivative. While the total number of generated partial derivatives is known to be linear in the size of the initial regular expression, no results can be found in the literature regarding the size of the largest partial derivative. We study this problem w. r. t. two metrics (height and size of regular expressions), and show that the former increases by at most one, while the latter is quadratic in the size of the regular expression. Surprisingly, these results also hold with shuffle.

Keywords

partial derivatives, regular expressions with shuffle, rewriting-based runtime verification, space complexity

1. Introduction

Derivatives of regular expressions were introduced by Brzozowski [1, 2] to provide a very compact operational semantics for them, and to derive an algorithm for generating equivalent deterministic finite automata (DFA). A first contribution of our work is to show that derivatives can be defined by means of a labeled transition system between regular expressions, where labels are the alphabet symbols. Given a fixed symbol, at each step there is always a unique transition, therefore the system defines a deterministic automaton where the initial state is an initial regular expression e, and all other states are regular expressions that are derivatives obtained from e. Such an automaton is provably equivalent to e, if the set of its final states is defined as the set of all the corresponding regular expressions which recognize the empty word. The main issue with this approach is that the number of syntactically different derivatives of e is generally unbounded, hence, to obtain an equivalent DFA, a finite quotient algebra of states modulo a suitable congruence has to be considered.

Inspired by Brzozowski's work, Antimirov [3] defined partial derivatives of regular expressions to define an algorithm that generates, from regular expressions, equivalent non-deterministic finite automata (NFA) with a limited number of states. The main breakthrough at the root of Antimirov's work is that the number of syntactically distinct partial derivatives of e is provably linear in the size of e. Therefore, a construction analogous to that proposed by Brzozowski can be used with syntactic equality without considering any weaker notion of congruence. However, the corresponding automaton is, in general non-deterministic, because of the transition system defining partial derivatives: given a fixed symbol, at each step there can be zero or more transitions.

ICTCS 2025: Italian Conference on Theoretical Computer Science, September 10–12, 2025, Pescara, Italy
*Corresponding author.
davide.ancona@unige.it (D. Ancona); angelo.ferrando@unimore.it (A. Ferrando)
0000-0002-6297-2011 (D. Ancona); 0000-0002-8711-4670 (A. Ferrando)

While Antimirov's work addresses the classical problem of word recognition in regular languages, here we focus on runtime verification (RV) [4] of simple properties expressible with regular expressions.

RV is a technique used to check the trace of events generated during a single execution of the System Under Scrutiny (SUS). This is done with monitors automatically generated from specifications that describe the correct behavior of the SUS (suitably instrumented to produce the trace to be analyzed).

RV is complementary to both formal verification and testing: like formal methods, it is based on specification languages, although it is not exhaustive; like testing, it is scalable and suitable for real-world systems and complex properties. Differently from testing, RV can detect errors even in non-deterministic systems [5, 6, 7, 8], and therefore represents a valuable complement to testing approaches [9].

When properties to be verified can be defined with regular expressions, words are finite traces of monitorable events, which define the alphabet of the language, and the monitor is the finite automaton generated from the regular expression defining the correct traces. However, this approach may be unfeasible because of the intractable number of states of the generated automaton, even for NFAs.

This typically happens when the properties to be verified consist of sub-traces of mutually independent events, which are allowed to interleave during the execution of the SUS. Consider, for instance, the challenge of verifying that files are properly opened and closed, in a system able to handle multiple files independently. In such cases, extending regular expressions with the shuffle operator [10] can significantly reduce the size of specifications and enhance clarity. Moreover, by using partial derivatives, it becomes possible to avoid generating automata with an intractable number of states, and adopt, instead, a rewriting-based approach [11, 10] to RV, where only a single partial derivative needs to be considered at each reduction step.

This raises the question of the space complexity of the largest partial derivative that can be generated from an initial regular expression. Indeed, while the total number of generated partial derivatives is known to be linear in the size of the initial regular expression, no results can be found in the literature regarding the size of the largest partial derivative.

In this paper, we investigate this problem with respect to two different metrics for regular expressions: their height and their total number of nodes when viewed as trees. In particular, we show that the height of the largest partial derivative can increase by at most one, while the number of nodes in the largest partial derivative is bounded by n^2, where n is the number of nodes of the initial regular expression. Surprisingly, these results still hold when regular expressions are extended with shuffle.

To this aim, we propose a proof methodology based on the definition of a function that, given a regular expression, returns an upper bound of the increment w.r.t. a specific metric for all its partial derivatives. In this way, an invariant can be established on single rewriting steps and directly extended to multiple steps, to derive the expected space complexity results. This proof methodology allows us to keep the same proof structure for all main results, and reuse parts of the proofs.

The paper is structured as follows: Section 2 introduces the basic notion of derivatives of regular expressions. Section 3 defines partial derivatives and proves the space complexity results w. r. t. the height and the size of expressions. Section 4 extends regular expressions with the shuffle operator, and show that the property proved in section 3 still hold. Finally, section 5 draws conclusions and some directions for future work. All major proofs can be found in the companion technical report [12].

2. Derivatives of regular expressions

This section provides the basic notions on derivatives, by assuming familiarity with regular expressions.

Let Σ denote a given non-empty finite set of symbols, called *alphabet*. We call *word* an element of Σ^*, and denote with λ the empty word. Furthermore, $\mathtt{a} : w$ denotes the word where $\mathtt{a}$ is the first symbol, and w the rest of the word, while $w \cdot w'$ denotes word concatenation. The set RE of regular expressions over Σ is inductively defined by the following grammar:

$$e \quad ::= \quad \mathtt{0} \mid \epsilon \mid \mathtt{a} \mid e_0 e_1 \mid e_0 + e_1 \mid e* \quad \text{with } \mathtt{a} \in \Sigma$$

We use the standard precedence rules: the Kleene star operator $e*$ has higher precedence than concatenation $e_0 e_1$, which in turn has higher precedence than union $e_0 + e_1$.

Definition 2.1 introduces the standard operators on regular languages, and definition 2.2 the standard semantics of regular expressions.

Definition 2.1. *For all* $L, L_0, L_1 \subseteq \Sigma^*$

$$L_1 \cdot L_2 = \{w_1 \cdot w_2 \mid w_1 \in L_1, w_2 \in L_2\} \quad L^0 = \{\lambda\}, L^{n+1} = L \cdot L^n \textit{ for all } n \geq 0 \quad L* = \bigcup_{n \geq 0} L^n$$

Definition 2.2.

$$[\![\mathtt{0}]\!] = \emptyset \quad [\![\epsilon]\!] = \{\lambda\} \quad [\![\mathtt{a}]\!] = \{\mathtt{a}\} \quad [\![e_0\, e_1]\!] = [\![e_0]\!] \cdot [\![e_1]\!] \quad [\![e_0 + e_1]\!] = [\![e_0]\!] \cup [\![e_1]\!] \quad [\![e_0 *]\!] = [\![e_0]\!] *$$

Note the difference between ϵ (a constant in the syntax of regular expressions) and λ (the empty word).

The derivative [1] of a language L w. r. t. a word $w \in \Sigma^*$, denoted by $w^{-1}(L)$, is defined as follows:

Definition 2.3. *For all* $L \subseteq \Sigma^*$, $w \in \Sigma^*$, $\quad w^{-1}(L) = \{w' \mid w \cdot w' \in L\}$.

Directly from definition 2.3 one can derive

$$w \in L \text{ iff } \lambda \in w^{-1}(L) \tag{1}$$

While the notion of derivative applies to any formal language, regular languages enjoy the following closure property: if L is regular, then $w^{-1}(L)$ is regular as well, for any word w. In other words, the derivative of a regular expression can be defined by another regular expression.

This property allows an operational and succinct definition for the derivatives of regular expressions. First, the derivative w. r. t. a single symbol is defined (see fig. 1), then the more general notion of derivative w. r. t. a word is derived by reflexive transitive closure (see definition 2.4).

Here we deliberately follow the style of a labelled transition system, defined by means of inference rules, to highlight two interesting related aspects:[1]

- derivatives provide an intuitive way to define a small-step semantics for regular expressions;
- a reduction step $e \xrightarrow{\mathtt{a}} e'$ corresponds to a transition step from the state represented by e to a state represented by e' with symbol $\mathtt{a}$, for an automaton which recognizes the language defined by e.

By using the notation of Antimirov [3], which is an adaptation of that introduced by Brzozowski [1], we have that $\mathtt{a}^{-1}(e_0) = e_1$ iff $e_0 \xrightarrow{\mathtt{a}} e_1$, and $w^{-1}(e_0) = e_1$ iff $e_0 \xrightarrow{w} e_1$.

The definition of $\nu(e)$ in fig. 1 deserves some comments: the intended meaning is that $\nu(e) = \epsilon$ iff $\lambda \in [\![e]\!]$, and, dually, $\nu(e) = \mathtt{0}$ iff $\lambda \notin [\![e]\!]$. The base cases of the definition are straightforward; thanks to the auxiliary functions and, or defined on ϵ and $\mathtt{0}$ in the corresponding tables, we can deduce that, as expected, $e_0\, e_1$ recognizes λ iff both e_0 and e_1 recognize λ, while $e_0 + e_1$ recognizes λ iff e_0 or e_1 recognizes λ.

The fact that $\nu(e)$ does not return a Boolean value was a deliberate choice of Brzozowski to define rule (CAT) in a more compact way: if $\nu(e_0) = \epsilon$, then $e_0\, e_1 \xrightarrow{\mathtt{a}} e_0' + \epsilon e_1'$, since in this case e_1 contributes to the derivative. Otherwise, $e_0\, e_1 \xrightarrow{\mathtt{a}} e_0' + \mathtt{0} e_1'$, since e_1 does not contribute to the derivative.

Note that by definition of the rules, for all e and $\mathtt{a}$, there always exists a unique derivative of e w. r. t. $\mathtt{a}$. For instance, if $\mathtt{a} \neq \mathtt{b}$, $\mathtt{a} \neq \mathtt{c}$, then $\mathtt{ab}+\mathtt{ac} \xrightarrow{\mathtt{a}} (\epsilon\mathtt{b}+\mathtt{00})+(\epsilon\mathtt{c}+\mathtt{00})$ and $\mathtt{ab}+\mathtt{ac} \xrightarrow{\mathtt{b}} (\mathtt{0b}+\mathtt{0}\epsilon)+(\mathtt{0c}+\mathtt{00})$.

Definition 2.4 introduces the general notion of derivative w. r. t. a word w, by computing the reflexive transitive closure of the one-step relation defined in fig. 1. As expected, the definition is by induction[2] on the length of w. The multiple-steps rewriting relation provides also the operational semantics of regular expressions w. r. t. derivatives, driven by eq. (1). In the view above, which considers the labelled transition system as a deterministic automaton, the definition corresponds to that of accepted language,

[1] To the best of our knowledge, no previous work has used this style to define the (partial) derivatives of regular expressions.

[2] It is worth noting that the computation of a derivative always "terminates" in a finite number of steps, because words have finite length.

$$\text{(EMPTY)}\frac{}{0 \xrightarrow{a} 0} \qquad \text{(EPS)}\frac{}{\epsilon \xrightarrow{a} 0} \qquad \text{(SYM-EQ)}\frac{}{a \xrightarrow{a} \epsilon} \qquad \text{(SYM-NEQ)}\frac{}{a \xrightarrow{b} 0}\ a \neq b$$

$$\text{(CAT)}\frac{e_0 \xrightarrow{a} e_0' \quad e_1 \xrightarrow{a} e_1'}{e_0 e_1 \xrightarrow{a} e_0' e_1 + \nu(e_0) e_1'} \qquad \text{(OR)}\frac{e_0 \xrightarrow{a} e_0' \quad e_1 \xrightarrow{a} e_1'}{e_0 + e_1 \xrightarrow{a} e_0' + e_1'} \qquad \text{(STAR)}\frac{e \xrightarrow{a} e'}{e* \xrightarrow{a} e' e*}$$

$\nu(\epsilon) = \nu(e_0*) = \epsilon$
$\nu(0) = \nu(a) = 0$
$\nu(e_0 e_1) = \nu(e_0)$ and $\nu(e_1)$
$\nu(e_0 + e_1) = \nu(e_0)$ or $\nu(e_1)$

and	0	ϵ
0	0	0
ϵ	0	ϵ

or	0	ϵ
0	0	ϵ
ϵ	ϵ	ϵ

Figure 1: Transition system defining the derivatives of regular expressions

$$\text{(SYM)}\frac{}{a \xrightarrow{a} \epsilon} \qquad \text{(L-CAT)}\frac{e_0 \xrightarrow{a} e_0'}{e_0 e_1 \xrightarrow{a} e_0' e_1} \qquad \text{(R-CAT)}\frac{e_1 \xrightarrow{a} e_1'}{e_0 e_1 \xrightarrow{a} e_1'}\ \nu(e_0) = \epsilon$$

$$\text{(L-OR)}\frac{e_0 \xrightarrow{a} e_0'}{e_0 + e_1 \xrightarrow{a} e_0'} \qquad \text{(R-OR)}\frac{e_1 \xrightarrow{a} e_1'}{e_0 + e_1 \xrightarrow{a} e_1'} \qquad \text{(STAR)}\frac{e \xrightarrow{a} e'}{e* \xrightarrow{a} e' e*}$$

Figure 2: Transition system defining the partial derivatives of regular expressions

where the set of final states is the set of all derivatives e' s. t. $\nu(e') = \epsilon$. It is straightforward to prove that the set of all syntactically distinct derivatives of a regular expression e is generally unbounded, and so is their size. Hence, the underlying automaton is infinite, unless some finite quotient algebra of states modulo a suitable congruence is considered.

Definition 2.4. *For all* $e, e_0, e_1 \in RE$, $a \in \Sigma$, $w \in \Sigma^*$

$$e \xrightarrow{\lambda} e \qquad e_0 \xrightarrow{a:w} e_1 \textit{ iff } e_0 \xrightarrow{a} e \textit{ and } e \xrightarrow{w} e_1$$
$$[\![e]\!]_{\rightarrow} = \{w \in \Sigma^* \mid \textit{there exists } e' \in RE \textit{ s.t. } e \xrightarrow{w} e' \textit{ and } \nu(e') = \epsilon\}$$

Theorem 2.5 establishes the results proved by Brzozowski [1]: the definitions of $\nu(e)$ and $e \xrightarrow{w} e'$ are sound w. r. t. their intended semantics, and the operational semantics of regular expressions is equivalent to the standard one.

Theorem 2.5. *For all* $e, e' \in RE$, $w \in \Sigma^*$

$$\nu(e) = \epsilon \textit{ iff } \lambda \in [\![e]\!] \qquad e \xrightarrow{w} e' \textit{ implies } w^{-1}([\![e]\!]) = [\![e']\!] \qquad [\![e]\!]_{\rightarrow} = [\![e]\!]$$

3. Partial derivatives of regular expressions

Figure 2 contains the transition rules defining the partial derivatives of regular expressions w. r. t. a single symbol. The definition follows the work of Antimirov [3], but as happens for the definition of derivatives, our presentation is based on a labelled transition system. More importantly, here we deliberately define a non-deterministic system, where a single transition step yields a single partial derivative among all possible ones, while in Antimirov's definition the whole set of partial derivatives is computed.[3] With our approach, the labelled transition system directly induces an NFA equivalent to the initial regular expression, and the main differences between derivatives and partial derivatives are highlighted, as explained below.

[3]This is for efficiency reasons, though the obtained states are still used by Antimirov to generate an NFA.

Rules (CAT) and (OR) in fig. 1 are split in fig. 2 in the two possibly overlapping[4] rules (L-CAT), (R-CAT) and (L-OR), (R-OR), respectively. For instance, according to Antimirov's definition[5], the partial derivative of $\mathtt{ab}+\mathtt{ac}$ w. r. t. symbol $\mathtt{a}$ returns the set of regular expressions $\{\epsilon\mathtt{b}, \epsilon\mathtt{c}\}$ (that is, $\delta_{\mathtt{a}}(\mathtt{ab}+\mathtt{ac}) = \{\epsilon\mathtt{b}, \epsilon\mathtt{c}\}$ with Antimirov's notation). This corresponds to the fact that there exist two possible transition steps from $\mathtt{ab}+\mathtt{ac}$ labeled with $\mathtt{a}$, namely, $\mathtt{ab}+\mathtt{ac} \xrightarrow{\mathtt{a}} \epsilon\mathtt{b}$ and $\mathtt{ab}+\mathtt{ac} \xrightarrow{\mathtt{a}} \epsilon\mathtt{c}$.

Another difference with derivatives is that in fig. 2 there are no rules returning the empty derivative. Therefore, there are cases where the partial derivative of a regular expression w. r. t. a symbol (or word) does not exist. This is useful when partial derivatives are used for RV, because in this way errors can be detected with no further checks on the partial derivative: if no moves can be taken, then the partial derivative is empty and the trace of events (that is, the word) considered so far cannot be the prefix of any correct trace, therefore a violation of the specification can be reported.

Finally, a major difference between derivatives and partial derivatives lies in the fact that the set of all syntactically distinct partial derivatives of a given regular expression e is bounded by the size of e. This means that, after a single partial derivative is generated at each rewriting step, there is no need to simplify it in order to keep the size of partial derivatives bounded.

Definition 3.1 is dual to Definition 2.4.

Definition 3.1. *For all* $e, e_0, e_1 \in RE$, $\mathtt{a} \in \Sigma$, $w \in \Sigma^*$

$$e \xrightarrow{\lambda}\!\!\!\!\!\!\rightharpoonup e \qquad e_0 \xrightarrow{\mathtt{a}:w}\!\!\!\!\!\!\rightharpoonup e_1 \textit{ iff } e_0 \xrightarrow{\mathtt{a}}\!\!\!\!\!\!\rightharpoonup e \textit{ and } e \xrightarrow{w}\!\!\!\!\!\!\rightharpoonup e_1$$
$$[\![e]\!]_{\rightharpoonup} = \{w \in \Sigma^* \mid \textit{there exists } e' \in RE \textit{ s.t. } e \xrightarrow{w}\!\!\!\!\!\!\rightharpoonup e' \textit{ and } \nu(e') = \epsilon\}$$

The claims below are dual to theorem 2.5, and directly follow from the results of Antimirov [3].

Let $\delta_w(e)$ denotes the set of all partial derivatives of $e \in RE$ w. r. t. $w \in \Sigma^*$:

$$\delta_w(e) = \{e' \in RE \mid e \xrightarrow{w}\!\!\!\!\!\!\rightharpoonup e'\}.$$

Theorem 3.2. *For all* $e \in RE$, $w^{-1}([\![e]\!]) = \bigcup_{e' \in \delta_w(e)} [\![e']\!]$ *and* $[\![e]\!]_{\rightharpoonup} = [\![e]\!]_{\rightarrow}$.

Corollary 3.3. *For all* $e \in RE$, $[\![e]\!]_{\rightharpoonup} = [\![e]\!]$.

The following theorem is based as well on the results proved by Antimirov [3].

Theorem 3.4. *For all* $e \in RE$, $\delta_w(e)$ *is bounded.*

3.1. Height of partial derivatives

In this section we investigate the upper bound on the height of the partial derivatives of a given regular expression, defined in a standard way as follows:

$$|\epsilon| = |\mathtt{a}| = 0 \qquad |e_0 e_1| = |e_0 + e_1| = \max(|e_0|, |e_1|) + 1 \qquad |e*| = |e| + 1$$

Since the height of the partial derivative of an expression e w. r. t. a word w both depends on the shape of e and on w, we need to reason on any possible e and w. Moreover, we aim at providing a general proof methodology which can be followed to prove results when considering also shuffle and the size of expressions.

To show a couple of examples let us consider the following reduction steps which compute the partial derivatives of $\mathtt{a}*\mathtt{b}*$ w. r. t. $\mathtt{ab}$ and $\mathtt{bb}$:

$$\mathtt{a}*\mathtt{b}* \xrightarrow{\mathtt{a}}\!\!\!\!\!\!\rightharpoonup (\epsilon\mathtt{a}*)\mathtt{b}* \xrightarrow{\mathtt{b}}\!\!\!\!\!\!\rightharpoonup \epsilon\mathtt{b}* \qquad \mathtt{a}*\mathtt{b}* \xrightarrow{\mathtt{b}}\!\!\!\!\!\!\rightharpoonup \epsilon\mathtt{b}* \xrightarrow{\mathtt{b}}\!\!\!\!\!\!\rightharpoonup \epsilon\mathtt{b}*$$
$$|\mathtt{a}*\mathtt{b}*| = 2 \qquad |(\epsilon\mathtt{a}*)\mathtt{b}*| = 3 \qquad |\epsilon\mathtt{b}*| = 2$$

[4] In the sense that for some regular expressions and symbols, both are applicable.

[5] See [3], definition 2.8.

In the first example the height increases by one after the first step and decreases by one after the second, while in the second example the height of the expressions is unchanged.

What we are going to prove are the following main properties:

1. the height of the partial derivative of a regular expression w. r. t. a symbol can increase **at most by one**;
2. the height of the partial derivative of a partial derivative w. r. t. a non-empty word, **cannot** increase;
3. from 1. and 2. and from the definition of partial derivative, one can deduce that if $e \xrightarrow{w} e'$, then $|e'| \leq |e| + 1$.

Although these properties can be proved in a direct way, they must be adapted when the size of expressions is considered. Furthermore, they do not hold in this stronger form, when shuffle is added. Therefore we provide a more general proof methodology which works for both metrics on expressions, and can be adapted to the case of shuffle, where weaker claims hold.

In order to do that, we define the function $\Delta_{\max}(e)$ which returns an upper bound for $|e'| - |e|$, where $e \xrightarrow{w} e'$, with $e' \in RE$, $w \in \Sigma^*$. That is, if $M_e \stackrel{\text{def.}}{=} \max\{|e'| - |e| \mid e \xrightarrow{w} e', e' \in RE, w \in \Sigma^*\}$, then a sound definition of $\Delta_{\max}(e)$ has to satisfy the constraint $\Delta_{\max}(e) \geq M_e$. Hence, $|e| + \Delta_{\max}(e)$ provides an upper bound for the height of all partial derivatives of e: $|e| + \Delta_{\max}(e) \geq \max\{|e'| \mid e \xrightarrow{w} e', e' \in RE, w \in \Sigma^*\}$.

Definition 3.5. *The function $\Delta_{\max}(e)$ is recursively defined as follows:*

$$\begin{array}{ll} \Delta_{\max}(\mathtt{a}) = \Delta_{\max}(\epsilon) = 0 & \Delta_{\max}(e_0\, e_1) = \mathit{geq}(e_0, e_1) \cdot \Delta_{\max}(e_0) \\ \Delta_{\max}(e_0 + e_1) = 0 & \Delta_{\max}(e\,*) = 1 \end{array}$$

$$\textit{where } \mathit{geq}(e_0, e_1) = \begin{cases} 1, & \textit{if } |e_0| \geq |e_1| \\ 0, & \textit{if } |e_0| < |e_1| \end{cases}$$

Before commenting the definition of $\Delta_{\max}(e)$, we introduce the following straightforward lemma.

Lemma 3.6. *For all $e \in RE$, $0 \leq \Delta_{\max}(e) \leq 1$.*

Proof. Directly by induction on the definition of $\Delta_{\max}(e)$. □

Because $e \xrightarrow{\lambda} e$, $0 \in \{|e'| - |e| \mid e \xrightarrow{w} e', e' \in RE, w \in \Sigma^*\}$ and the constraint $0 \leq \Delta_{\max}(e)$ must always be met. The constraint $\Delta_{\max}(e) \leq 1$ corresponds to what we want to prove in item 3 above.

The definition of $\Delta_{\max}(e)$ is driven by the reduction rules in fig. 2 and by the property in item 3. We comment only the non-trivial cases: if $e = e_0 + e_1$, then we expect $|e_i'| \leq |e_i| + 1$, $i = 0, 1$, hence $|e_i'| \leq |e|$ and $\Delta_{\max}(e) = 0$; if $e = e_0\,*$, then we expect $|e_0'| \leq |e_0| + 1$, hence $|e_0' e_0\,*| \leq |e| + 1$ and $\Delta_{\max}(e) = 1$. Finally, the definition for $e = e_0 e_1$ requires more care: if rule (L-CAT) is applied, then $|e_0' e_1| = |e| + 1$, but only when $|e_0'| = |e_0| + 1$ and $|e_0| \geq |e_1|$, otherwise $|e_0' e_1| \leq |e|$; if rule (R-CAT) is applied, then $|e_1'| \leq |e|$, as happens for rule (R-OR). Therefore $\Delta_{\max}(e) = 1$ iff $\Delta_{\max}(e_0) = 1$ and $|e_0| \geq |e_1|$, that is, $\mathit{geq}(e_0, e_1) = 1$. In all other cases, that is $\Delta_{\max}(e_0) = 0$ or $\mathit{geq}(e_0, e_1) = 0$, $\Delta_{\max}(e) = 0$.

The following theorem proves the soundness of the definition of $\Delta_{\max}$ w. r. t. its intended meaning. As a consequence, the height of the derivative of an expression e w. r. t. a symbol is always bounded by $|e| + 1$.

Theorem 3.7. *For all $e, e' \in RE$, $\mathtt{a} \in \Sigma$, if $e \xrightarrow{\mathtt{a}} e'$, then $|e'| \leq |e| + \Delta_{\max}(e)$.*

The following corollary can be directly derived from theorem 3.7 and lemma 3.6.

Corollary 3.8. *For all $e, e' \in RE$, and $\mathtt{a} \in \Sigma$, if $e \xrightarrow{\mathtt{a}} e'$, then $|e'| \leq |e| + 1$.*

Theorem 3.7 shows that the height of the derivative of an expression e w. r. t. a symbol is bounded by $|e| + 1$. The following results prove a strong property: the height of the partial derivative of a partial derivative cannot increase; that is, after the first reduction step, the height of the derivative of an expression e w. r. t. a symbol is bounded by $|e|$. Therefore, one can conclude that the height of the derivative of an expression e w. r. t. an arbitrary word (not just a symbol) is bounded by $|e| + 1$.

Lemma 3.9. *For all $e, e' \in RE$, and $\mathtt{a} \in \Sigma$, if $e \xrightarrow{\mathtt{a}} e'$, then $\Delta_{\max}(e') = 0$.*

Given lemma 3.9, the following corollary may look useless, but it provides a general form of invariant which can be proved also for all other considered cases, where the strong claim of lemma 3.9 no longer holds.

Corollary 3.10. *For all $e, e' \in RE$, $\mathtt{a} \in \Sigma$, if $e \xrightarrow{\mathtt{a}} e'$, then $|e'| \leq |e| + \Delta_{\max}(e) - \Delta_{\max}(e')$.*

Proof. A direct consequence of theorem 3.7 and lemma 3.9. □

To provide intuition for the invariant of corollary 3.10, let us consider the following equivalent form:

$$|e'| + \Delta_{\max}(e') \leq |e| + \Delta_{\max}(e) \tag{2}$$

As noted in the previous page, $|e'| + \Delta_{\max}(e')$ and $|e| + \Delta_{\max}(e)$ define the upper bound of the height of all partial derivatives of e' and e, respectively. But e' is a partial derivative of e because $e \xrightarrow{\mathtt{a}} e'$, hence, by transitivity, the set P' of all partial derivatives of e' is a subset of all partial derivatives P of e.

The following theorem shows that the invariant property of corollary 3.10 can be extended to derivatives w. r. t. any words.

Theorem 3.11. *For all $e, e' \in RE$, and $w \in \Sigma^*$, if $e \xrightarrow{w} e'$, then $|e'| \leq |e| + \Delta_{\max}(e) - \Delta_{\max}(e')$.*

Proof. By induction on the length of w.

base case: if $w = \lambda$, then by definition $e' = e$, hence $|e'| = |e|$ and $\Delta_{\max}(e) = \Delta_{\max}(e')$, therefore $|e'| = |e| \leq |e| + \Delta_{\max}(e) - \Delta_{\max}(e')$.

inductive step: if $w = \mathtt{a} : w'$ for some $\mathtt{a} \in \Sigma$, $w' \in \Sigma^*$, then by definition there exists $e'' \in RE$ s.t. $e \xrightarrow{\mathtt{a}} e''$ and $e'' \xrightarrow{w'} e'$. By corollary 3.10 $|e''| \leq |e| + \Delta_{\max}(e) - \Delta_{\max}(e'')$ and by inductive hypothesis $|e'| \leq |e''| + \Delta_{\max}(e'') - \Delta_{\max}(e')$. Therefore by transitivity, $|e'| \leq |e''| + \Delta_{\max}(e'') - \Delta_{\max}(e') \leq |e| + \Delta_{\max}(e) - \Delta_{\max}(e'') + \Delta_{\max}(e'') - \Delta_{\max}(e') = |e| + \Delta_{\max}(e) - \Delta_{\max}(e')$.

□

The proof of theorem 3.11 is independent from the definition of $\Delta_{\max}$, providing that the invariant of corollary 3.10 holds. Therefore, the crucial point is the definition of the function returning the upper bound of the increment of a partial derivation.

The result on the space complexity of derivatives can be directly derived from theorem 3.11 and lemma 3.6.

Corollary 3.12. *For all $e, e' \in RE$, and $w \in \Sigma^*$, if $e \xrightarrow{w} e'$, then $|e'| \leq |e| + 1$.*

Proof. By theorem 3.11 $|e'| \leq |e| + \Delta_{\max}(e) - \Delta_{\max}(e')$. By lemma 3.6 $\Delta_{\max}(e) - \Delta_{\max}(e') \leq 1$, hence $|e'| \leq |e| + 1$. □

3.2. Size of partial derivatives

In this section we investigate the upper bound on the size of the partial derivatives of a given regular expression, defined in a standard way as follows:

$$\|\epsilon\| = \|\mathtt{a}\| = 1 \qquad \|e_0\, e_1\| = \|e_0 + e_1\| = \|e_0\| + \|e_1\| + 1 \qquad \|e\,*\| = \|e\| + 1$$

The results of section 3.1 must be adapted to the less trivial case of the size of expressions. Indeed, while for the height the upper bound of the increment is the constant 1, for the size this value depends on the square of the initial regular expression.

For instance, let us consider the following reduction steps which compute the partial derivative of $\mathtt{a}***$ w. r. t. $\mathtt{aa}$:

$$\mathtt{a}*** \xrightarrow{\mathtt{a}} ((\epsilon\mathtt{a}*)\mathtt{a}**)\mathtt{a}*** \xrightarrow{\mathtt{a}} ((\epsilon\mathtt{a}*)\mathtt{a}**)\mathtt{a}***$$
$$\|\mathtt{a}***\| = 4 \qquad \|((\epsilon\mathtt{a}*)\mathtt{a}**)\mathtt{a}***\| = 13$$

After the first step the size increases by 9 and remains unchanged after the second step. For expressions of the general shape $e = \mathtt{a}*\ldots*$ it can be proved by induction on $n = \|e\| \geq 2$ that if $e \xrightarrow{\mathtt{a}} e'$, then $\|e'\| = \|e\| + \frac{n^2+n}{2} - 1$.

As another example, let us consider the following reduction steps which compute the partial derivative of $(\mathtt{a}*\mathtt{b}*)\mathtt{c}*$ w. r. t. $\mathtt{abc}$:

$$\mathtt{ab}** \xrightarrow{\mathtt{a}} \epsilon\mathtt{b}** \xrightarrow{\mathtt{b}} (\epsilon\mathtt{b}*)\mathtt{b}**$$
$$\|\mathtt{ab}**\| = 5 \qquad \|\epsilon\mathtt{b}**\| = 5 \qquad \|(\epsilon\mathtt{b}*)\mathtt{b}**\| = 8$$

After the first step the size does not change, while increases by 3 after the second step. This example shows that the strong claim of lemma 3.9 cannot hold for $\|\ \|$.

We follow the same methodology as in section 3.1 and define the function $\eta_{\max}(e)$, analogous to $\Delta_{\max}(e)$, for the size of e.

Definition 3.13. *The function $\eta_{\max}(e)$ is recursively defined as follows:*

$$\begin{aligned}
&\eta_{\max}(\mathtt{a}) = \eta_{\max}(\epsilon) = 0\\
&\eta_{\max}(e_0\, e_1) = \max(\eta_{\max}(e_0), \eta_{\max}(e_1) - \|e_0\| - 1)\\
&\eta_{\max}(e_0 + e_1) = \max(\eta_{\max}(e_0) - \|e_1\| - 1, \eta_{\max}(e_1) - \|e_0\| - 1, 0)\\
&\eta_{\max}(e\,*) = \|e\| + \eta_{\max}(e) + 1
\end{aligned}$$

The following lemma is analogous to lemma 3.6.

Lemma 3.14. *For all $e \in RE$, $0 \leq \eta_{\max}(e) \leq \|e\|^2$.*

As for $\Delta_{\max}(e)$, the definition of $\eta_{\max}(e)$ is driven by the reduction rules in fig. 2, and by the definition of size. We comment only the non-trivial cases: if $e = e_0 + e_1$, then the partial derivative can be obtained by applying either rule (L-OR) or (R-OR). For the former rule, the partial derivative of e_0 can increase of at most $\eta_{\max}(e_0)$, but all the nodes of e_1 and the $+$ operator are removed, which corresponds to $\eta_{\max}(e_0) - \|e_1\| - 1$. The reasoning for the latter rule is symmetric. Then the maximum between these two values has to be considered to return a valid upper bound; furthermore, the bound must be non-negative for the same reason explained for $\Delta_{\max}$. If $e = e_0\,*$, then the new term e' needs to be considered, whose size is bounded by $\|e\| + \eta_{\max}(e)$. Furthermore, the concatenation operator is added, hence the function returns $\|e\| + \eta_{\max}(e) + 1$. Finally, in the definition for $e = e_0\, e_1$ rule (R-CAT) is managed as rule (R-OR) (or (L-OR)). In rule (L-CAT) e_0 is replaced by e_0', while e_1 and concatenation are kept, hence the upper bound of the increment is $\eta_{\max}(e_0)$. As for $e_0 + e_1$ the maximum needs to be considered.

Although the analogous of lemma 3.9 cannot be proved for $\eta_{\max}$, the invariant of corollary 3.10 holds, and can be proved directly.

Theorem 3.15. *For all $e, e' \in RE$, $\mathtt{a} \in \Sigma$, if $e \xrightarrow{\mathtt{a}} e'$, then $\|e'\| \leq \|e\| + \eta_{\max}(e) - \eta_{\max}(e')$.*

Having proved the invariant of theorem 3.15, we can derive the main result analogously as done in section 3.1.

Theorem 3.16. *For all $e, e' \in RE$, and $w \in \Sigma^*$, if $e \xrightarrow{w}_{\searrow} e'$, then $\|e'\| \leq \|e\| + \eta_{\max}(e) - \eta_{\max}(e')$.*

Proof. See theorem 3.11 □

The result on the space complexity of derivatives directly follows from theorem 3.16 and lemma 3.14.

Corollary 3.17. *For all $e, e' \in RE$, and $w \in \Sigma^*$, if $e \xrightarrow{w}_{\searrow} e'$, then $\|e'\| \leq \|e\| + \|e\|^2$. Hence $\|e'\|$ is $O(\|e\|^2)$.*

Proof. By theorem 3.16 $\|e'\| \leq \|e\| + \eta_{\max}(e) - \eta_{\max}(e')$. By lemma 3.6 $\eta_{\max}(e) - \eta_{\max}(e') \leq \eta_{\max}(e) \leq \|e\|^2$, hence $\|e'\| \leq \|e\| + \|e\|^2$. □

4. Regular expressions with shuffle

4.1. Basic definitions

In this section we extend the syntax of regular expressions by adding the shuffle operator[6] $e_0 \,||\, e_1$ whose semantics is defined as follows by extending definition 2.1 and definition 2.2:

$$\lambda \,||\, w = w \,||\, \lambda = \{w\} \qquad (\mathtt{a}_1 w_1) \,||\, (\mathtt{a}_2 w_2) = \{\mathtt{a}_1\} \cdot (w_1 \,||\, (\mathtt{a}_2 w_2)) \cup \{\mathtt{a}_2\} \cdot ((\mathtt{a}_1 w_1) \,||\, w_2)$$
$$L_1 \,||\, L_2 = \bigcup_{w_1 \in L_1, w_2 \in L_2} (w_1 \,||\, w_2) \qquad [\![e_0 \,||\, e_1]\!] = [\![e_0]\!] \,||\, [\![e_1]\!]$$

We denote with $RE+$ the set of regular expressions extended with the shuffle operator.

Although it is well-known that the shuffle operator does not increase the abstract expressive power of regular expressions, in practice it is still very useful in RV to write much more compact and clear specifications when correct system behaviors can be described as independently interleaved event traces. Indeed, it has been proved [13, 14] that there exists a family of regular expressions e with shuffle for which the equivalent NFA needs at least $2^{\|e\|}$ states.

Let us consider for instance the distinct events o_n, a_n, and c_n, with the meaning "file n has been opened", "accessed", and "closed", respectively, where $n = 1, 2$ is the corresponding file descriptor. If the SUS is allowed to manage the two files independently, then a specification for the correct use of them can be defined quite concisely by the regular expression $o_1 a_1 {*} c_1 \,||\, o_2 a_2 {*} c_2$.

If, for simplicity, we assume that files can be accessed only once, and we generalize over the number of files, then the specification becomes the regular expression $o_1 a_1 c_1 \,||\, \ldots \,||\, o_n a_n c_n$ over the alphabet of $3n$ distinct symbols $\{o_1, a_1, c_1, \ldots, o_n, a_n, c_n\}$. For such an expression, an equivalent NFA must have at least 4^n states.

$$\text{(SHF)}\ \frac{e_0 \xrightarrow{\mathtt{a}} e_0' \quad e_1 \xrightarrow{\mathtt{a}} e_1'}{e_0 \,||\, e_1 \xrightarrow{\mathtt{a}} (e_0' \,||\, e_1) + (e_0 \,||\, e_1')} \qquad \nu(e_0 \,||\, e_1) = \nu(e_0) \text{ and } \nu(e_1)$$

$$\text{(L-SHF)}\ \frac{e_0 \xrightarrow{\mathtt{a}}_{\searrow} e_0'}{e_0 \,||\, e_1 \xrightarrow{\mathtt{a}}_{\searrow} e_0' \,||\, e_1} \qquad \text{(R-SHF)}\ \frac{e_1 \xrightarrow{\mathtt{a}}_{\searrow} e_1'}{e_0 \,||\, e_1 \xrightarrow{\mathtt{a}}_{\searrow} e_0 \,||\, e_1'}$$

Figure 3: Transition rules defining the derivative and the partial derivatives for the shuffle operator

The transition rules defining the derivative and the partial derivatives for the shuffle operator can be found in fig. 3. They all correspond to the intuition that events can be interleaved; moreover, the empty trace is contained in $e_0 \,||\, e_1$ iff it is contained in e_0 and e_1.

[6] In the examples we assume that the shuffle has lower precedence than the concatenation and union operator.

As happens for the other operators, also with the shuffle the derivative is always defined, while this is not the case for partial derivatives. For instance, if we assume $\mathtt{a}_2 \neq \mathtt{a}_0$ and $\mathtt{a}_2 \neq \mathtt{a}_1$, then $\mathtt{a}_0 \,||\, \mathtt{a}_1 \xrightarrow{\mathtt{a}_2} (\mathtt{0} \,||\, \mathtt{a}_1) + (\mathtt{a}_0 \,||\, \mathtt{0})$, and $\nu((\mathtt{0} \,||\, \mathtt{a}_1) + (\mathtt{a}_0 \,||\, \mathtt{0})) = \mathtt{0}$, but there exists no e s.t. $\mathtt{a}_0 \,||\, \mathtt{a}_1 \xrightarrow{\mathtt{a}_2} e$.

Theorems 2.5 and 3.2 still hold along with corollary 3.3, when $RE+$ is considered.

4.2. Height of partial derivatives with the shuffle operator

We extend the result of theorem 3.11 and corollary 3.12 to the case of the shuffle operators.

Unfortunately the proofs are more challenging because the claim of lemma 3.9 no longer holds: There exist partial derivatives e s. t. $\Delta_{\max}(e) > 0$. As a counter example, let us consider the following reduction steps computing the partial derivatives of $(\epsilon \,||\, \mathtt{a}*)(\mathtt{b} \,||\, \mathtt{a}*)$ w. r. t. $\mathtt{aba}$:

$$e_0 \xrightarrow{\mathtt{a}} e_1 \xrightarrow{\mathtt{b}} e_2 \xrightarrow{\mathtt{a}} e_3$$
$$e_0 = (\epsilon \,||\, \mathtt{a}*)(\mathtt{b} \,||\, \mathtt{a}*) \qquad e_1 = (\epsilon \,||\, \epsilon\mathtt{a}*)(\mathtt{b} \,||\, \mathtt{a}*) \qquad e_2 = \epsilon \,||\, \mathtt{a}* \qquad e_3 = \epsilon \,||\, \epsilon\mathtt{a}*$$

We have $|e_1| = |e_0| + 1$ (first reduction step), but also $|e_3| = |e_2| + 1$ (third reduction step). Therefore, necessarily $\Delta_{\max}(e_2) > 0$, to ensure that eq. (2) holds.

To prove the extended versions of the results of section 3 we first extend the definition of $\Delta_{\max}$ given in definition 3.5 with the case for the shuffle.

$$\Delta_{\max}(e_0 \,||\, e_1) = \max(geq(e_0, e_1) \cdot \Delta_{\max}(e_0), geq(e_1, e_0) \cdot \Delta_{\max}(e_1))$$

Before explaining the definition of $\Delta_{\max}$ above, we note that the claim of lemma 3.6 holds also for $RE+$ and can still be proved by induction on the definition of $\Delta_{\max}$: $0 \leq \Delta_{\max}(e) \leq 1$. Consequently, if the height of one sub-expression e_i is strictly greater than the other e_{1-i}, then only the partial derivative of e_i can contribute to the increment of the height of the partial derivative of $e_0 \,||\, e_1$, because increments are always bounded by 1. Note that $\max(\Delta_{\max}(e_i), 0) = \Delta_{\max}(e_i)$, since $\Delta_{\max}(e_i) \geq 0$, therefore $\Delta_{\max}(e_0 \,||\, e_1) = \Delta_{\max}(e_i)$, if $|e_i| > |e_{1-i}|$ for $i = 0, 1$.

If $|e_0| = |e_1|$, then both the derivatives of e_0 and e_1 can contribute to the increment of the height of the partial derivative of $e_0 \,||\, e_1$. Since (L-SHF) or (R-SHF) can be non-deterministically applied, the maximum increment $\max(\Delta_{\max}(e_0), \Delta_{\max}(e_1))$ needs to be considered.

The correctness of the definition of $\Delta_{\max}(e_0 \,||\, e_1)$ is still ensured by the claim of theorem 3.7 extended to $RE+$.

Proof of theorem 3.7 extended to $RE+$

For all $e, e' \in RE+$, $\mathtt{a} \in \Sigma$, if $e \xrightarrow{\mathtt{a}} e'$, then $|e'| \leq |e| + \Delta_{\max}(e)$.

Proof. The structure of the proof is the same, but the two new rules for the shuffle operator need to be considered.

- rule (L-SHF): By inductive hypothesis $|e_0'| \leq |e_0| + \Delta_{\max}(e_0)$. We distinguish two cases:
  - $|e_0| \geq |e_1|$: $|e_0' \,||\, e_1| \stackrel{\text{def}}{=} \max(|e_0'|, |e_1|) + 1 \leq \max(|e_0| + \Delta_{\max}(e_0), |e_1|) + 1 \leq \max(|e_0|, |e_1|) + 1 + \Delta_{\max}(e_0) \leq \max(|e_0|, |e_1|) + 1 + \Delta_{\max}(e_0 \,||\, e_1) = |e_0 \,||\, e_1| + \Delta_{\max}(e_0 \,||\, e_1)$, where inequalities are derived from the inductive hypothesis, the definition of $\max$, $|\,|$, $\Delta_{\max}$, and geq, the assumption $|e_0| \geq |e_1|$, and lemma 3.6.
  - $|e_0| < |e_1|$ (that is, $|e_0| + \Delta_{\max}(e_0) \leq |e_1|$ by lemma 3.6): $|e_0' \,||\, e_1| \stackrel{\text{def}}{=} \max(|e_0'|, |e_1|) + 1 \leq \max(|e_0| + \Delta_{\max}(e_0), |e_1|) + 1 = |e_0 \,||\, e_1| \leq |e_0 \,||\, e_1| + \Delta_{\max}(e_0 \,||\, e_1)$, where inequalities are derived from the inductive hypothesis, the definition of $\max$, $|\,|$, $\Delta_{\max}$, and geq, the assumption $|e_0| < |e_1|$, and lemma 3.6.
- rule (r-shf) is symmetric to rule (l-shf).

□

Since both lemma 3.6 and theorem 3.7 hold for $RE+$, corollary 3.8 holds for $RE+$ as well.

To prove for $RE+$ the invariant defined by eq. (2), we modularize the proof by introducing two lemmas.

The first lemma is a weaker version of lemma 3.9.

Lemma 4.1. *For all $e, e' \in RE+$, and $\mathtt{a} \in \Sigma$, if $e \xrightarrow{\mathtt{a}} e'$ and $|e'| = |e| + 1$, then $\Delta_{\max}(e') = 0$.*

The second lemma establishes an invariant on $\Delta_{\max}$ when the height of the derivatives does not change.

Lemma 4.2. *For all $e, e' \in RE+$, and $\mathtt{a} \in \Sigma$, if $e \xrightarrow{\mathtt{a}} e'$ and $|e'| = |e|$, then $\Delta_{\max}(e') \leq \Delta_{\max}(e)$.*

Theorem 4.3. *For all $e, e' \in RE+$, $\mathtt{a} \in \Sigma$, if $e \xrightarrow{\mathtt{a}} e'$, then $|e'| \leq |e| + \Delta_{\max}(e) - \Delta_{\max}(e')$.*

From theorem 4.3 and lemma 3.6 (extended to $RE+$) we can extend for free theorem 3.11 and corollary 3.12 to $RE+$.

Theorem 4.4. *For all $e, e' \in RE+$, and $w \in \Sigma^*$, if $e \xrightarrow{w} e'$, then $|e'| \leq |e| + \Delta_{\max}(e) - \Delta_{\max}(e')$.*

Corollary 4.5. *For all $e, e' \in RE+$, and $w \in \Sigma^*$, if $e \xrightarrow{w} e'$, then $|e'| \leq |e| + 1$.*

4.3. Size of partial derivatives with the shuffle operator

Similar as done in the previous section, we extend the result of theorem 3.16 and corollary 3.17 to the case of the shuffle operator in the case of the size $\|e\|$ of a regular expression e.

Also for the size, the introduction of shuffle does not affect the space complexity of partial derivatives. However, in this case the proofs can be extended in an easier way, although the definition of $\eta_{\max}(e_0 \mid\mid e_1)$ requires some care.

Indeed, one would be tempted to consider the following (incorrect) definition:

$$\eta_{\max}(e_0 \mid\mid e_1) = \max(\eta_{\max}(e_0), \eta_{\max}(e_1)) \qquad \text{(incorrect definition)}$$

Unfortunately, this definition verifies only the weaker inequality $\|e\| \leq \|e_0 \mid\mid e_1\| + \eta_{\max}(e_0 \mid\mid e_1)$, if $e_0 \mid\mid e_1 \xrightarrow{\mathtt{a}} e$, but not $\|e\| \leq \|e_0 \mid\mid e_1\| + \eta_{\max}(e_0 \mid\mid e_1) - \eta_{\max}(e)$.

To show this, let us consider the following reduction step computing the partial derivative of $\mathtt{a}* \mid\mid \mathtt{b}*$ w. r. t. $\mathtt{a}$:

$$\mathtt{a}* \mid\mid \mathtt{b}* \xrightarrow{\mathtt{a}} \epsilon\mathtt{a}* \mid\mid \mathtt{b}*$$
$$\|\mathtt{a}* \mid\mid \mathtt{b}*\| = 5 \qquad \eta_{\max}(\mathtt{a}* \mid\mid \mathtt{b}*) = 2 \qquad \|\epsilon\mathtt{a}* \mid\mid \mathtt{b}*\| = 7 \qquad \eta_{\max}(\epsilon\mathtt{a}* \mid\mid \mathtt{b}*) = 2$$

We have $\|\epsilon\mathtt{a}* \mid\mid \mathtt{b}*\| = 7 \leq 5 + 2 = \|\mathtt{a}* \mid\mid \mathtt{b}*\| + \eta_{\max}(\mathtt{a}* \mid\mid \mathtt{b}*)$, but $\|\epsilon\mathtt{a}* \mid\mid \mathtt{b}*\| = 7 \not\leq 5 + 2 - 2 = \|\mathtt{a}* \mid\mid \mathtt{b}*\| + \eta_{\max}(\mathtt{a}* \mid\mid \mathtt{b}*) - \eta_{\max}(\epsilon\mathtt{a}* \mid\mid \mathtt{b}*)$.

To guarantee the invariant $\|e'\| \leq \|e\| + \eta_{\max}(e) - \eta_{\max}(e')$, the following definition has to be considered:

$$\eta_{\max}(e_0 \mid\mid e_1) = \eta_{\max}(e_0) + \eta_{\max}(e_1) \qquad \text{(correct definition)}$$

This new definition takes into account increments due to multiple reduction steps. In this way, we obtain $\|\epsilon\mathtt{a}* \mid\mid \mathtt{b}*\| = 7 \leq 5 + 4 - 2 = \|\mathtt{a}* \mid\mid \mathtt{b}*\| + \eta_{\max}(\mathtt{a}* \mid\mid \mathtt{b}*) - \eta_{\max}(\epsilon\mathtt{a}* \mid\mid \mathtt{b}*)$.

The correctness of the definition of $\eta_{\max}(e_0 \mid\mid e_1)$ is ensured by the claim of theorem 3.15 extended to $RE+$.

Proof of theorem 3.15 extended to $RE+$

For all $e, e' \in RE+$, $\mathtt{a} \in \Sigma$, if $e \xrightarrow{\mathtt{a}} e'$, then $\|e'\| \leq |e| + \eta_{\max}(e) - \eta_{\max}(e')$.

Proof. The structure of the proof is the same, but the two new rules for the shuffle operator need to be considered.

- rule (L-SHF): By inductive hypothesis $\|e'_0\| \leq \|e_0\| + \eta_{\max}(e_0) - \eta_{\max}(e'_0)$.
  We have $\|e'_0 \mid\mid e_1\| = \|e'_0\| + \|e_1\| + 1 \leq \|e_0\| + \eta_{\max}(e_0) - \eta_{\max}(e'_0) + \|e_1\| + 1 = \|e_0 \mid\mid e_1\| + \eta_{\max}(e_0 \mid\mid e_1) - \eta_{\max}(e_1) - \eta_{\max}(e'_0 \mid\mid e_1) + \eta_{\max}(e_1) = \|e_0 \mid\mid e_1\| + \eta_{\max}(e_0 \mid\mid e_1) - \eta_{\max}(e'_0 \mid\mid e_1)$, by the inductive hypothesis, the definition of $\| \ \|$, and $\eta_{\max}$.
- rule (r-shf) is symmetric to rule (l-shf).

□

From theorem 3.15 and lemma 3.14 (extended to $RE+$) we can extend for free theorem 3.16 and corollary 3.17 to $RE+$.

Theorem 4.6. *For all $e, e' \in RE+$, and $w \in \Sigma^*$, if $e \xrightarrow{w} e'$, then $\|e'\| \leq \|e\| + \eta_{\max}(e) - \eta_{\max}(e')$.*

Corollary 4.7. *For all $e, e' \in RE+$, and $w \in \Sigma^*$, if $e \xrightarrow{w} e'$, then $\|e'\| \leq \|e\| + \|e\|^2$. Hence $\|e'\|$ is $O(\|e\|^2)$.*

5. Conclusion

We have explored the space complexity of partial derivatives of regular expressions in the context of RV with regular expressions extended with the shuffle operator. While the size of the set of syntactically distinct partial derivatives of a regular expression e is known to be linear in the size of e, no bounds on the size of the largest derivative had been previously investigated. We fill this gap by analyzing the height and size of partial derivatives and establishing upper bounds for both metrics.

We have shown that the height of any partial derivative of a regular expression increases by at most one, and that this property still holds when the shuffle operator is considered. Furthermore, we proved that the size of the largest partial derivative is bounded quadratically in the size of the original expression. This quadratic bound also holds in the presence of shuffle, despite with this operator there exist regular expressions whose equivalent NFAs exhibit an exponential explosion of the number of states.

Our approach is based on a general proof methodology that defines functions to compute upper bounds on the increase in height and size of the partial derivatives. These functions allows us to establish an invariant that can be generalized to multiple rewriting steps, enabling a modular and reusable proof structure. This methodology allowed us to seamlessly extend our results to a more complex operator like shuffle.

Our results support the practical use of partial derivatives in rewriting-based RV, ensuring that the memory usage remains manageable even in the worst case. Moreover, they show that the shuffle operator is useful for writing short and readable specifications without incurring high computational costs.

Short-term future work includes the extension of our analysis to other improvements of the expressive power of regular expressions, for instance with the use of additional operators or parameterized specifications.

Declaration on Generative AI

During the preparation of this work, the authors used ChatGPT in order to: Grammar and spelling check, Paraphrase and reword. After using this tool/service, the authors reviewed and edited the content as needed and take full responsibility for the publication's content.

References

[1] J. A. Brzozowski, Derivatives of Regular Expressions, J. ACM 11 (4) (1964) 481–494.

[2] S. Owens, J. Reppy, A. Turon, Regular-expression derivatives re-examined, Journal of Functional Programming 19 (2) (2009) 173–190.

[3] V. Antimirov, Partial derivatives of regular expressions and finite automaton constructions, Theoretical Computer Science 155 (2) (1996) 291–319, ISSN 0304-3975.

[4] M. Leucker, C. Schallhart, A brief account of runtime verification, J. Log. Algebr. Methods Program. 78 (5) (2009) 293–303.

[5] K. Havelund, G. Roşu, An Overview of the Runtime Verification Tool Java PathExplorer, Form. Methods Syst. Des. 24 (2) (2004) 189–215.

[6] S. Sharma, G. Gopalakrishnan, E. Mercer, J. Holt, MCC: A runtime verification tool for MCAPI user applications, in: Proceedings of 9th International Conference on Formal Methods in Computer-Aided Design (FMCAD'09), IEEE, 41–44, 2009.

[7] D. Ancona, L. Franceschini, G. Delzanno, M. Leotta, M. Ribaudo, F. Ricca, Towards Runtime Monitoring of Node.js and Its Application to the Internet of Things, in: D. Pianini, G. Salvaneschi (Eds.), Proceedings of the 1st Workshop on Architectures, Languages and Paradigms for IoT (ALP4IoT'17), vol. 264 of *EPTCS*, 27–42, 2017.

[8] F. Schiavio, H. Sun, D. Bonetta, A. Rosà, W. Binder, NodeMOP: runtime verification for Node.js applications, in: Proceedings of the 34th ACM/SIGAPP Symposium on Applied Computing (SAC'19), ACM, 1794–1801, 2019.

[9] V. Besnard, M. Huet, S. Bivolarov, N. Saadi, G. Cornard, AMT: A Runtime Verification Tool of Video Streams, in: P. Katsaros, L. Nenzi (Eds.), Proceedings of the 23rd International Conference on Runtime Verification (RV'23), vol. 14245 of *LNCS*, Springer, 315–326, 2023.

[10] D. Ancona, L. Franceschini, A. Ferrando, V. Mascardi, RML: Theory and practice of a domain specific language for runtime verification, Sci. Comput. Program. 205 (2021) 102610.

[11] G. Rosu, K. Havelund, Rewriting-Based Techniques for Runtime Verification, Automated Software Engineering 12 (2) (2005) 151–197.

[12] D. Ancona, A. Ferrando, On The Space Complexity of Partial Derivatives of Regular Expressions with Shuffle, Tech. Rep., URL https://arxiv.org/, 2025.

[13] S. Broda, A. Machiavelo, N. Moreira, R. Reis, Automata for regular expressions with shuffle, Information and Computation 259 (2018) 162–173, ISSN 0890-5401.

[14] A. Mayer, L. Stockmeyer, The Complexity of Word Problems - This Time with Interleaving, Information and Computation 115 (2) (1994) 293–311, ISSN 0890-5401.