Manipulation of Regular Expressions Using Derivatives: an Overview*

Nelma Moreira and Rogério Reis

CMUP & DCC, Faculdade de Ciências da Universidade do Porto,
Rua do Campo Alegre, 4169-007 Porto, Portugal
{nelma.moreira,rogerio.reis}@fc.up.pt

Abstract. The notions of derivative and partial derivative of regular expressions revealed themselves to be very powerful and have been successfully extended to many other formal language classes and algebraic structures. Although the undisputed elegance of this formalism, its efficient practical use is still a challenging research topic. Here we give a brief historical overview and summarise some of these aspects.

1 Preliminares

Regular expressions are the common choice to represent regular languages due to their succinctness and clear syntax. Deterministic finite automata are an excellent representation for testing equivalence, containment, or membership, as these problems are easily solved for this model. However, minimal deterministic finite automata (DFA) can be exponentially larger than the associated regular expression, while corresponding nondeterministic finite automata (NFA) are only linearly larger. The computational and descriptional complexity of regular expressions and of conversions to and from finite automata are well studied. Good surveys on the subject are [40, 39]. In recent years, the average size of different NFA constructions from regular expressions were studied using the framework of analytic combinatorics [63, 14, 17]. For the average case, the uniform distribution on the set of regular expressions is considered although that does not imply a uniform representation of regular languages. In this survey, we focus on the derivative and partial derivative based constructions. First, we recall some basic notions and fix notation.

Given an alphabet $\Sigma = \{\sigma_1, \ldots, \sigma_k\}$ of size $k \geq 1$, a *language* L is a subset of the free monoid $\Sigma^\star$. The *left-quotient* of a language L by a word $w \in \Sigma^\star$, is the language $w^{-1}L = \{x \mid wx \in L\}$.

The set $\mathcal{R}_k$ of (standard) *regular expressions* r over Σ is composed by $\emptyset$ plus the expressions defined by the following context-free grammar:

$$\mathsf{r} := \varepsilon \mid \sigma_1 \mid \cdots \mid \sigma_k \mid (\mathsf{r} + \mathsf{r}) \mid (\mathsf{r} \odot \mathsf{r}) \mid (\mathsf{r}^\star), \tag{1}$$

where the symbol $\odot$ is normally omitted and represents concatenation. The (regular) *language* represented by an expression $\mathsf{r} \in \mathcal{R}_k$ is denoted by $\mathcal{L}(\mathsf{r})$ and

* Research supported by CMUP through FCT project UIDB/00144/2021.

is inductively defined as follows for $\mathsf{r}, \mathsf{r}' \in \mathcal{R}_k$: $\mathcal{L}(\emptyset) = \emptyset$, $\mathcal{L}(\varepsilon) = \{\varepsilon\}$, $\mathcal{L}(\sigma) = \{\sigma\}$, $\mathcal{L}(\mathsf{r} + \mathsf{r}') = \mathcal{L}(\mathsf{r}) \cup \mathcal{L}(\mathsf{r}')$, $\mathcal{L}(\mathsf{r} \odot \mathsf{r}') = \mathcal{L}(\mathsf{r})\mathcal{L}(\mathsf{r}') = \{\, wv \mid w \in \mathcal{L}(\mathsf{r}) \wedge v \in \mathcal{L}(\mathsf{r}') \,\}$, and $\mathcal{L}(\mathsf{r}^\star) = \mathcal{L}(\mathsf{r})^\star = \bigcup_{n\in\mathbb{N}}(\mathcal{L}(\mathsf{r})^n)$. For the *size* of a regular expression r, denoted by $\|\mathsf{r}\|$, we consider the size of its syntactic tree, i.e., the number of symbols in r, not counting parentheses but including $\odot$. The *alphabetic size* of r, denoted by $|\mathsf{r}|_\Sigma$, is just the number of alphabetic symbols in r. We define $\varepsilon(\mathsf{r}) = \varepsilon$ if $\varepsilon \in \mathcal{L}(\mathsf{r})$, and $\varepsilon(\mathsf{r}) = \emptyset$, otherwise. The function $\varepsilon()$ is easily defined inductively in the structure of r. Two expressions r and s are *equivalent* if their languages are the same, and we write $\mathsf{r} = \mathsf{s}$. With this interpretation, the algebraic structure $(\mathcal{R}_k, +, \cdot, \emptyset, \varepsilon)$ is a idempotent semiring that with $\star$ forms a Kleene algebra.

A *nondeterministic finite automaton* (NFA) is a quintuple $\mathcal{A} = \langle Q, \Sigma, \delta, I, F\rangle$ where Q is a finite set of states, Σ is a finite alphabet, $I \subseteq Q$ is the set of initial states, $F \subseteq Q$ is the set of final states, and $\delta : Q \times \Sigma \to 2^Q$ is the transition function. The *language accepted* by $\mathcal{A}$ is $\mathcal{L}(\mathcal{A}) = \{w \in \Sigma^\star \mid \delta(I, w) \cap F \neq \emptyset\}$. When $I = \{q_0\}$, we use $I = q_0$. If $|I| = 1$ and $|\delta(q, \sigma)| \leq 1$, for all $q \in Q, \sigma \in \Sigma$, $\mathcal{A}$ is *deterministic* (DFA). For a DFA $\mathcal{A}$, $w \in \mathcal{L}(\mathcal{A})$ if $\delta(q_0, w) \in F$. Two NFAs $\mathcal{A}$ and $\mathcal{A}'$ are *equivalent* if their languages are the same. An automaton $\mathcal{A}$ is equivalent to a regular expression r if $\mathcal{L}(\mathcal{A}) = \mathcal{L}(\mathsf{r})$. We can convert an NFA $\mathcal{A}$ into an equivalent DFA $\mathrm{D}(\mathcal{A})$ by the *determinisation* operation D, using the well-known subset construction. An equivalence relation $\equiv$ on Q is *right invariant* w.r.t. an NFA $\mathcal{A}$ if and only if $\equiv \subseteq (Q - F)^2 \cup F^2$ and $\forall p, q \in Q, \sigma \in \Sigma$, if $p \equiv q$ then $\forall p' \in \delta(p, \sigma)\ \exists q' \in \delta(q, \sigma)$ such that $p' \equiv q'$. The *quotient automaton* $\mathcal{A}/\!\equiv$ is given by $A/\!\equiv\ = \langle Q/\!\equiv, \Sigma, \delta/\!\equiv, I/\!\equiv, F/\!\equiv\rangle$, where $\delta/\!\equiv([p], \sigma) = \{\, [q] \mid q \in \delta(p, \sigma) \,\} = \delta(p, \sigma)/\!\equiv$. It is easy to see that $\mathcal{L}\,(\mathcal{A}/\!\equiv) = \mathcal{L}(\mathcal{A})$.

2 Derivatives

In 1962, Janusz Brzozowski introduced the notion of derivative of a regular expression in his Ph.D. thesis *Regular Expression Techniques for Sequential Circuits* [22]. Based on nerve nets of McCulloch and Pitts [52], in 1956 Kleene [44] showed the equivalence of finite automata and regular expressions. Brzozowski proposed regular expressions as a simple formalism for describing the behaviour of sequential circuits as opposed to use directly finite automata (state graphs), as regular expressions are in general more readable. A theory of regular expressions was developed for the conversion of expressions to finite automata and vice-versa. Methods for converting finite automata into regular expression were already known [61, 53] but a simple method was presented, nowadays known as the *state elimination method* [21]. For the conversion from regular expressions into finite automata, there existed already methods such as the McNaugthon-Yamada automaton ($\mathcal{A}_{\mathrm{MY}}$) [53]. Brzozowski defined a deterministic finite automaton equivalent to a regular expression using the notion of derivative [22, 25]. The *derivative* by $\sigma \in \Sigma$ of a regular expression r is a regular expression

$\mathrm{d}_\sigma(\mathsf{r})$, inductively defined by:

$$\begin{aligned} \mathrm{d}_\sigma(\emptyset) &= \mathrm{d}_\sigma(\varepsilon) = \emptyset, \\ \mathrm{d}_\sigma(\sigma') &= \begin{cases} \{\varepsilon\} & \text{if } \sigma' = \sigma, \\ \emptyset & \text{otherwise,} \end{cases} \end{aligned} \qquad \begin{aligned} \mathrm{d}_\sigma(\mathsf{r}+\mathsf{r}') &= \mathrm{d}_\sigma(\mathsf{r}) + \mathrm{d}_\sigma(\mathsf{r}'), \\ \mathrm{d}_\sigma(\mathsf{r}\mathsf{r}') &= \begin{cases} \mathrm{d}_\sigma(\mathsf{r})\mathsf{r}' & \text{if } \varepsilon(\mathsf{r}) = \emptyset, \\ \mathrm{d}_\sigma(\mathsf{r})\mathsf{r}' + \mathrm{d}_\sigma(\mathsf{r}') & \text{otherwise,} \end{cases} \\ \mathrm{d}_\sigma(\mathsf{r}^\star) &= \mathrm{d}_\sigma(\mathsf{r})\mathsf{r}^\star. \end{aligned} \tag{2}$$

This notion can be extended to words: $\mathrm{d}_\varepsilon(\mathsf{r}) = \mathsf{r}$ and $\mathrm{d}_{\sigma w}(\mathsf{r}) = \mathrm{d}_w(\mathrm{d}_\sigma(\mathsf{r}))$. The language of $\mathrm{d}_w(\mathsf{r})$ is $\mathcal{L}(\mathrm{d}_w(\mathsf{r})) = \{\, x \mid wx \in \mathcal{L}(\mathsf{r}) \,\} = w^{-1}\mathcal{L}(\mathsf{r})$. The set of all derivatives of r, $\{\, \mathrm{d}_w(\mathsf{r}) \mid w \in \Sigma^\star \,\}$, may not be finite. For finiteness, Brzozowski considered the quotient of that set modulo some regular expression equivalences, namely the associativity, commutativity, and idempotence of $+$ (ACI) and the following rules: $\mathsf{r}\varepsilon = \varepsilon\mathsf{r} = \mathsf{r}$, $\emptyset\mathsf{r} = \mathsf{r}\emptyset = \emptyset$, and $\emptyset + \mathsf{r} = \mathsf{r} + \emptyset = \mathsf{r}$.[1] Let $\mathcal{D}(\mathsf{r})$ be the resulting set. The *Brzozowski's automaton* for r is the DFA defined as follows

$$\mathcal{A}_\mathrm{B}(\mathsf{r}) = \langle \mathcal{D}(\mathsf{r}), \Sigma, \delta_\mathrm{B}, [\mathsf{r}], F_\mathrm{B} \rangle, \tag{3}$$

where $F_\mathrm{B} = \{\, [\mathsf{r}'] \in \mathcal{D}(\mathsf{r}) \mid \varepsilon(\mathsf{r}') = \varepsilon \,\}$, and $\delta_\mathrm{B}([\mathsf{r}'], \sigma) = [\mathrm{d}_\sigma(\mathsf{r}')]$, for all $[\mathsf{r}] \in \mathcal{D}(\mathsf{r})$ and $\sigma \in \Sigma$. The proof that $\mathcal{D}(\mathsf{r})$ is finite constitute one of the first results on state complexity bounds. A language L is recognised by a finite automaton if and only if it has a finite number of left-quotients [62], and that number is the state complexity of L. The language of a derivative is a left-quotient, but two derivatives may represent the same language. Known upper bounds for the state complexity of several operations can be obtained using derivatives by tightening the bounds of $|\mathcal{D}(\mathsf{r})|$ [23]. Both the size $|\mathcal{D}(\mathsf{r})|$ and the size of the elements of $\mathcal{D}(\mathsf{r})$ can grow exponentially with $\|\mathsf{r}\|$.

Derivatives can be used to decide problems such as the word membership, universality, or equivalence of regular expressions, avoiding the automaton construction.

Membership A word $w \in \mathcal{L}(\mathsf{r})$ if and only in $\varepsilon(\mathrm{d}_w(\mathsf{r})) = \varepsilon$. This method can be extended to nonregular languages.

Universality A regular expression r represents $\Sigma^\star$ if and only if for all $[\mathsf{r}'] \in \mathcal{D}(\mathsf{r})$, $\varepsilon([\mathsf{r}']) = \varepsilon$.

Equivalence The correctness of the Brzozowski's automaton relies in the following equivalence

$$\mathsf{r} = \varepsilon(\mathsf{r}) + \sum_{\sigma \in \Sigma} \sigma\, \mathrm{d}_\sigma(\mathsf{r}), \tag{4}$$

If $\mathcal{D}(\mathsf{r}) = \{\mathsf{r} = \mathsf{r}_1, \ldots, \mathsf{r}_n\}$, then the following system of equations is satisfied

$$\mathsf{r}_i = \varepsilon(\mathsf{r}_i) + \sum_{j=1}^{k} \sigma_j \mathsf{r}_{i,j}, \tag{5}$$

[1] The necessity of these equalities was pointed out by Salomaa [72].

where $\mathsf{r}_{i,j}$ is $[\mathrm{d}_{\sigma_j}(\mathsf{r}_i)]$. One has that $\mathsf{r} = \mathsf{s}$ if and only if $\varepsilon(\mathsf{r}) = \varepsilon(\mathsf{s})$ and $[\mathrm{d}_{\sigma_j}(\mathsf{r})] = [\mathrm{d}_{\sigma_j}(\mathsf{s})]$ for all $\sigma_j \in \Sigma$. Below we present a refutation method such that testing the equivalence of the two expressions corresponds to an iterated process of testing the equivalence of their derivatives [5, 2].

```
equivP(r,s):
    S = {(r,s)}
    H = ∅
    while (r,s) = POP(S):
        if ε(r) ≠ ε(s): return False
        PUSH(H, (r,s))
        for σ ∈ Σ:
            (r',s') = ([d_σ(r)],[d_σ(s)])
            if (r',s') ∉ H: PUSH(S,(r',s'))
    return True
```

This method is related to the Hopcroft and Karp's algorithm for testing the equivalence of two deterministic finite automata that avoids their minimisation [41, 3]. Ginzburg [36] argues that the above method is *cumbersome* due to the computation of derivatives and equivalence classes, and presents a similar method but using NFAs.

Regular expressions can be extended to include any Boolean operation $\oplus$ on regular languages. Brzozowski defined $\mathrm{d}_\sigma(\oplus(\mathsf{r}_1, \ldots, \mathsf{r}_n)) = \oplus(\mathrm{d}_\sigma(\mathsf{r}_1), \ldots, \mathrm{d}_\sigma(\mathsf{r}_n))$, for $\sigma \in \Sigma$. In particular, $\mathrm{d}_\sigma(\mathsf{r} \cap \mathsf{r}') = \mathrm{d}_\sigma(\mathsf{r}) \cap \mathrm{d}_\sigma(\mathsf{r}')$ and $\mathrm{d}_\sigma(\neg\mathsf{r}) = \neg\,\mathrm{d}_\sigma(\mathsf{r})$. Again, he proved that an extended regular expression has a finite number of derivatives modulo some equivalences and thus a DFA could be constructed, solving a problem stated in [53].

In the next decades, derivatives were useful in several algebraic characterizations, for instance [72, 33, 24, 48]; inspired conversions from expressions to automata, such as the Thompson automaton ($\mathcal{A}_{\varepsilon\text{-T}}$) [77]; or were the based of regular expression equivalence tests [36, 58]. However, for practical applications, manipulation methods based directly on regular expressions were thought much more inefficients than the ones based on the conversion of regular expressions to finite automata. Also the fact that regular expressions needed to be considered modulo ACI was a disadvantage. One exception is Berry and Sethi's method of constructing the McNaughton-Yamada DFA ($\mathcal{A}_{\text{MY}}$) using derivatives [9].

3 Partial Derivatives

In 1966, Boris G. Mirkin [57] presented an algorithm for constructing an NFA that is a nondeterministic counterpart of Brzozowski's automaton. Mirkin considered a system of equations as (5) but where the $\mathsf{r}_{i,j}$ can be sums of expressions. The solution of the system leads to a nondeterminitic automaton construction. Given a regular expression r_0, a set of expressions $\pi(\mathsf{r}_0) = \{\mathsf{r}_1, \ldots, \mathsf{r}_n\}$ is a *support* of r_0 if for each $\mathsf{r}_i \in \{\mathsf{r}_0\} \cup \pi(\mathsf{r}_0)$ the equation (5) holds where each $\mathsf{r}_{i,j}$ is a (possibly empty) sum of elements in $\pi(\mathsf{r}_0)$. The set $\{\mathsf{r}_0\} \cup \pi(\mathsf{r}_0)$ is a *pre-base* of r_0. Mirkin

proved that a support of $\mathsf{r} \in \mathcal{R}_k$ can be inductively defined as follows

$$\begin{array}{ll} \pi(\emptyset) = \emptyset, & \pi(\mathsf{r}+\mathsf{s}) = \pi(\mathsf{r}) \cup \pi(\mathsf{s}), \\ \pi(\varepsilon) = \emptyset, & \pi(\mathsf{rs}) = \pi(\mathsf{r})\mathsf{s} \cup \pi(\mathsf{s}), \\ \pi(\sigma) = \{\varepsilon\}, & \pi(\mathsf{r}^\star) = \pi(\mathsf{r})\mathsf{r}^\star, \end{array} \tag{6}$$

where, for any $S \subseteq \mathcal{R}_k$, we define $S\emptyset = \emptyset S = \emptyset$, $S\varepsilon = \varepsilon S = S$, and $S\mathsf{r}' = \{\, \mathsf{r}\mathsf{r}' \mid \mathsf{r} \in S \wedge \mathsf{r} \neq \varepsilon \,\} \cup \{\, \mathsf{r}' \mid \varepsilon \in S \,\}$ if $\mathsf{r}' \neq \emptyset, \varepsilon$ (and analogously for $\mathsf{r}'S$). It is easy to see that $|\pi(\mathsf{r})| \leq |\mathsf{r}|_\Sigma$ and thus a relatively small NFA can be constructed. Moreover, it follows also from Mirkin's proof that set of transitions of this automaton can be inductively defined [13, 19].

Almost thirty years later, and independently, Valentin Antimirov [4] introduced partial derivatives as a (non-deterministic) generalisation of derivatives and obtained an NFA construction, called the partial derivative automaton, $\mathcal{A}_{\mathrm{PD}}$ Champarnaud and Ziadi [27] proved that the Mirkin and Antimirov automaton constructions are equivalent. Essentially, Antimirov associates to a left-quotient of $\mathcal{L}(\mathsf{r})$ a set of regular expressions instead of a unique expression. For a regular expression $\mathsf{r} \in \mathcal{R}_k$ and a symbol $\sigma \in \Sigma$, the *set of partial derivatives by σ of r* is defined inductively as follows:

$$\begin{array}{ll} \partial_\sigma(\emptyset) = \partial_\sigma(\varepsilon) = \emptyset, & \partial_\sigma(\mathsf{r}+\mathsf{r}') = \partial_\sigma(\mathsf{r}) \cup \partial_\sigma(\mathsf{r}'), \\ \partial_\sigma(\sigma') = \begin{cases} \{\varepsilon\} & \text{if } \sigma' = \sigma, \\ \emptyset & \text{otherwise,} \end{cases} & \partial_\sigma(\mathsf{r}\mathsf{r}') = \partial_\sigma(\mathsf{r})\mathsf{r}' \cup \varepsilon(\mathsf{r})\partial_\sigma(\mathsf{r}'), \\ & \partial_\sigma(\mathsf{r}^\star) = \partial_\sigma(\mathsf{r})\mathsf{r}^\star, \end{array} \tag{7}$$

where for $S \subseteq \mathcal{R}_k$ and $\mathsf{r} \in \mathcal{R}_k$, we consider $S\mathsf{r}$ or $\mathsf{r}S$ as above. The set of partial derivatives by a word $w \in \Sigma^\star$ of $\mathsf{r} \in \mathcal{R}_k$ is inductively defined by $\partial_\varepsilon(\mathsf{r}) = \{\mathsf{r}\}$ and $\partial_{w\sigma}(\mathsf{r}) = \partial_\sigma(\partial_w(\mathsf{r}))$. We have that $\mathcal{L}(\mathrm{d}_w(\mathsf{r})) = \mathcal{L}(\partial_w(\mathsf{r})) = \bigcup_{\mathsf{r}' \in \partial_w(\mathsf{r})} \mathcal{L}(\mathsf{r}')$, for $w \in \Sigma^\star$. The set of all partial derivatives of r by nonempty words is $\partial^+(\mathsf{r}) = \bigcup_{w \in \Sigma^\star \setminus \{\varepsilon\}} \partial_w(\mathsf{r})$ and coincides with $\pi(\mathsf{r})$, i.e., $\partial^+(\mathsf{r}) = \pi(\mathsf{r})$ [27]. Equation (2) can be redefined as follows

$$\mathsf{r} = \varepsilon(\mathsf{r}) \cup \bigcup_{\sigma \in \Sigma} \sigma\partial_\sigma(\mathsf{r}), \tag{8}$$

where we denote the right-hand side by $\mathsf{L}_\varepsilon(\mathsf{r})$. This means that membership, universality, equivalence, and related decision problems can be solved, adapting the procedures given above, to sets of partial derivatives.

The *partial derivative automaton* of $\mathsf{r} \in \mathcal{R}_k$ is defined as

$$\mathcal{A}_{\mathrm{PD}}(\mathsf{r}) = \langle \mathsf{PD}(\mathsf{r}), \Sigma, \delta_{\mathrm{PD}}, \mathsf{r}, F_{\mathrm{PD}} \rangle,$$

where $\mathsf{PD}(\mathsf{r}) = \partial^+(\mathsf{r}) \cup \{\mathsf{r}\}$, $F_{\mathrm{PD}} = \{\, \mathsf{r}' \in \mathsf{PD}(\mathsf{r}) \mid \varepsilon(\mathsf{r}') = \varepsilon \,\}$, and $\delta_{\mathrm{PD}}(\mathsf{r}', \sigma) = \partial_\sigma(\mathsf{r}')$, for all $\mathsf{r}' \in \mathsf{PD}(\mathsf{r})$ and $\sigma \in \Sigma$. We have, $|\mathsf{PD}(\mathsf{r})| \leq |\mathsf{r}|_\Sigma + 1$.

Both Mirkin and Antimirov argued that the DFA obtained from $\mathcal{A}_{\mathrm{PD}}(\mathsf{r})$ by determinisation, $\mathrm{D}(\mathcal{A}_{\mathrm{PD}}(\mathsf{r}))$, has several advantages over $\mathcal{A}_{\mathrm{B}}(\mathsf{r})$: avoids the computation of a equivalence relation; has at most $2^{|\mathsf{r}|_\Sigma}$ states; each state of $\mathrm{D}(\mathcal{A}_{\mathrm{PD}}(\mathsf{r}))$ is a set of partial derivatives $\partial_w(\mathsf{r}) \subseteq \mathsf{PD}(\mathsf{r})$ and thus each of these sets could be defined using references to some elements of $\mathsf{PD}(\mathsf{r})$.

We note that $\mathrm{D}(\mathcal{A}_{\mathrm{PD}}(\mathsf{r}))$ is not isomorphic to $\mathcal{A}_{\mathrm{B}}$, i.e., $\mathrm{D}(\mathcal{A}_{\mathrm{PD}}(\mathsf{r})) \not\simeq \mathcal{A}_{\mathrm{B}}$. This is mainly due to the distributivity of the concatenation over a set of expressions in Equation (7) (see [64]). In [6, 11] it was shown that $\mathrm{D}(\mathcal{A}_{\mathrm{PD}}(\mathsf{r}))/\equiv_{\mathsf{L}_\varepsilon} \simeq \mathcal{A}_{\mathrm{B}}(\mathsf{r})/\equiv_{\mathsf{L}_\varepsilon}$ where $\equiv_{\mathsf{L}_\varepsilon}$ is a right-invariant equivalence relation w.r.t. these automata such that $S \equiv_{\mathsf{L}_\varepsilon} S'$ if and only if $\mathsf{L}_\varepsilon(S) = \mathsf{L}_\varepsilon(S')$, for $S, S' \subseteq \mathcal{R}_k$. The resulting quotient automaton can be directly obtained by the determinisation of yet another automaton construction based on partial derivatives, that we denote by $\mathcal{A}_{\overleftarrow{\mathrm{Pre}}}$. From Equation (8), the elements of $\mathsf{L}_\varepsilon(\mathsf{r})$ are either ε or expressions of the form $\sigma\mathsf{r}'$. Consider the function $\overleftarrow{\mathsf{p}}_w(\mathsf{r})$ on words $w \in \Sigma^\star$ defined inductively by $\overleftarrow{\mathsf{p}}_\varepsilon(\mathsf{r}) = \mathsf{L}_\varepsilon(\mathsf{r})$ and $\overleftarrow{\mathsf{p}}_{w\sigma}(\mathsf{r}) = \bigcup_{\sigma\mathsf{r}' \in \overleftarrow{\mathsf{p}}_w(\mathsf{r})} \mathsf{L}_\varepsilon(\mathsf{r}')$. It is immediate that $\mathcal{L}(\overleftarrow{\mathsf{p}}_w(\mathsf{r})) = w^{-1}\mathcal{L}(\mathsf{r})$. The automaton $\mathcal{A}_{\overleftarrow{\mathrm{Pre}}}(\mathsf{r})$ is a NFA equivalent to r defined by

$$\mathcal{A}_{\overleftarrow{\mathrm{Pre}}}(\mathsf{r}) = \langle \overleftarrow{\mathrm{Pre}}(\mathsf{r}), \Sigma, \delta_{\overleftarrow{\mathrm{Pre}}}, \mathsf{L}_\varepsilon(\mathsf{r}), \varepsilon \rangle,$$

where $\overleftarrow{\mathrm{Pre}}(\mathsf{r}) = \bigcup_{w\in\Sigma^\star} \overleftarrow{\mathsf{p}}_w(\mathsf{r})$ and $\delta_{\overleftarrow{\mathrm{Pre}}}(\mathsf{r}', \sigma) = \mathsf{L}_\varepsilon(\mathsf{r}'')$ if $\mathsf{r}' = \sigma\mathsf{r}''$, and $\delta_{\overleftarrow{\mathrm{Pre}}}(\mathsf{r}', \sigma) = \emptyset$, otherwise. Then , we have [11]

$$\mathrm{D}(\mathcal{A}_{\mathrm{PD}}(\mathsf{r}))/\equiv_{\mathsf{L}_\varepsilon} \simeq \mathcal{A}_{\mathrm{B}}(\mathsf{r})/\equiv_{\mathsf{L}_\varepsilon} \simeq \mathrm{D}(\mathcal{A}_{\overleftarrow{\mathrm{Pre}}}(\mathsf{r})).$$

This DFA is interesting because, it is the smallest among several deterministic automata constructions obtained from regular expressions, although not always the minimal [11].

For a language $L \subseteq \Sigma^\star$ and a word $w \in \Sigma^\star$ one can also define the *right-quotient* of L by w, as $Lw^{-1} = \{\, x \mid xw \in L \,\}$. The notions of derivative and partial-derivative can also be defined in this case, as well as the correspondent automata. However, that is tantmount to consider the left constructions in the double reverse, i.e., $\mathcal{A}(\mathsf{r}^{\mathrm{R}})^{\mathrm{R}}$ where R is the reversal operation. Of course, one has $\mathcal{L}(\mathcal{A}(\mathsf{r})) = \mathcal{L}(\mathcal{A}(\mathsf{r}^{\mathrm{R}})^{\mathrm{R}}) = \mathcal{L}(\mathsf{r})$. In particular, $\mathcal{A}_{\overleftarrow{\mathrm{Pre}}}(\mathsf{r}) \simeq \mathcal{A}_{\mathrm{Pre}}(\mathsf{r}^{\mathrm{R}})^{\mathrm{R}}$ where $\mathcal{A}_{\mathrm{Pre}}$ is the *prefix automaton* introduced by Yamamoto [78] and studied in [19, 11]. Broda et al [11] presented a taxonomy of conversions from regular expressions to finite automaton that includes the above ones and that are related with the position automaton, which we consider in the next section.

3.1 Position Automaton

The position automaton $\mathcal{A}_{\mathrm{POS}}$, introduced by Victor Glushkov [37] in 1961, permits us to convert a regular expression r into an equivalent NFA without ε-transitions. McNaughton-Yamada 1960's automaton [53], $\mathcal{A}_{\mathrm{MY}}$, corresponds to the determinisation of $\mathcal{A}_{\mathrm{POS}}$ and the construction is similar. Leiss inductive automaton construction [49] leads to the same automaton [30]. This automaton is also called *standard* as it has a unique initial state which is *non-returning* [70, 71]. Below we will see its connection with the partial derivative automaton, $\mathcal{A}_{\mathrm{PD}}$.

The states in the position automaton correspond to the positions of alphabetic symbols in r plus an additional initial state. Formally, given $\mathsf{r} \in \mathcal{R}_k$, one can mark each occurrence of a symbol $\sigma \in \Sigma$ with its position in r, considering reading it from left to right. The resulting regular expression is a

marked regular expression $\bar{\mathsf{r}}$ with all alphabetic symbols distinct. Then, a position $i \in [1, |\mathsf{r}|_\Sigma]$ corresponds to the symbol σ_i in $\bar{\mathsf{r}}$, and consequently to exactly one occurrence of σ in r. The same notation is used to remove the markings, i.e., $\overline{\bar{\mathsf{r}}} = \mathsf{r}$. Let $\mathsf{Pos}(\mathsf{r}) = \{1, 2, \ldots, |\mathsf{r}|_\Sigma\}$, and let $\mathsf{Pos}_0(\mathsf{r}) = \mathsf{Pos}(\mathsf{r}) \cup \{0\}$. To define the $\mathcal{A}_{\mathrm{POS}}(\mathsf{r})$ we consider the following sets: $\mathsf{First}(\mathsf{r}) = \{\, i \mid \sigma_i w \in \mathcal{L}(\bar{\mathsf{r}}) \,\}$, $\mathsf{Last}(\mathsf{r}) = \{\, i \mid w\sigma_i \in \mathcal{L}(\bar{\mathsf{r}}) \,\}$, and $\mathsf{Follow}(\mathsf{r}, i) = \{\, j \mid u\sigma_i\sigma_j v \in \mathcal{L}(\bar{\mathsf{r}}) \,\}$. Given a set $S \subseteq \mathsf{Pos}(\mathsf{r})$ and $\sigma \in \Sigma$, let $\mathsf{Select}(S, \sigma) = \{\, i \mid i \in S \wedge \overline{\sigma_i} = \sigma \,\}$. Then, the *position automaton* for r is

$$\mathcal{A}_{\mathrm{POS}}(\mathsf{r}) = \langle \mathsf{Pos}_0(\mathsf{r}), \Sigma, \delta_{\mathrm{POS}}, 0, \mathsf{Last}(\mathsf{r}) \cup \varepsilon(\mathsf{r})\{0\} \rangle,$$

where $\delta_{\mathrm{POS}}(i, \sigma) = \mathsf{Select}(\mathsf{Follow}(\mathsf{r}, i), \sigma)$.

Champarnaud and Ziadi [28] proved that $\mathcal{A}_{\mathrm{PD}}$ is a quotient of the position automaton $\mathcal{A}_{\mathrm{POS}}$ by the right-invariant equivalence relation $\equiv_c$, that we define in the following. Given a position i, for all $w \in \Sigma^\star_{\bar{\mathsf{r}}}$, $\partial_{w\sigma_i}(\bar{\mathsf{r}})$ is either empty or equal to the singleton $\{\mathsf{c}(\mathsf{r}, i)\}$, which element is called the i's c-continuation of $\bar{\mathsf{r}}$. For $i \in \mathsf{Pos}(\mathsf{r})$, c-continuations are inductively defined by: $\mathsf{c}(\emptyset, i) = \mathsf{c}(\varepsilon, i) = \emptyset$, and $\mathsf{c}(\sigma_i, i) = \varepsilon$. Now consider $\bar{\mathsf{r}}$ of the form $\mathsf{r}_1 + \mathsf{r}_2$, $\mathsf{r}_1\mathsf{r}_2$, or $\mathsf{r}_1^\star$. If i occurs in r_1, then $\mathsf{c}(\mathsf{r}_1 + \mathsf{r}_2, i) = \mathsf{c}(\mathsf{r}_1, i)$, $\mathsf{c}(\mathsf{r}_1\mathsf{r}_2, i) = \mathsf{c}(\mathsf{r}_1, i)\mathsf{r}_2$, and $\mathsf{c}(\mathsf{r}_1^\star, i) = \mathsf{c}(\mathsf{r}_1, i)\mathsf{r}_1^\star$. If i occurs in r_2, then $\mathsf{c}(\mathsf{r}_1 + \mathsf{r}_2, i) = \underline{\mathsf{c}(\mathsf{r}_1\mathsf{r}_2, i)} = \underline{\mathsf{c}(\mathsf{r}_2, i)}$. Considering $\mathsf{c}(\mathsf{r}, 0) = \bar{\mathsf{r}}$, for $i, j \in \mathsf{Pos}_0(\mathsf{r})$ we define $i \equiv_c j \Leftrightarrow \underline{\mathsf{c}(\mathsf{r}, i)} = \underline{\mathsf{c}(\mathsf{r}, j)}$.

Proposition 1 ([28]). *$\mathcal{A}_{\mathrm{PD}}(r) \simeq \mathcal{A}_{\mathrm{POS}}(r)/\equiv_c$.*

The proof of this proposition relies in the following relations

- $\partial^+(\bar{\mathsf{r}}) = \{\, \mathsf{c}(\mathsf{r}, i) \mid i \in \mathsf{Pos}(\mathsf{r}) \,\}$;
- $\partial_{\sigma_i}(\bar{\mathsf{r}}) = \{\mathsf{c}(\mathsf{r}, i)\} \iff i \in \mathsf{First}(\mathsf{r})$;
- $\varepsilon(\mathsf{c}(\mathsf{r}, i)) = \varepsilon \iff i \in \mathsf{Last}(\mathsf{r})$;
- $\mathsf{c}(\mathsf{r}, i) \in \partial_{\sigma_i}(\mathsf{c}(\mathsf{r}, j)) \iff i \in \mathsf{Follow}(\mathsf{r}, j)$.

From that, one has that $\underline{\mathcal{A}_{\mathrm{PD}}(\bar{\mathsf{r}})} \simeq \mathcal{A}_{\mathrm{POS}}(\bar{\mathsf{r}})/\equiv'_c$, where $i \equiv'_c j \Leftrightarrow \mathsf{c}(\mathsf{r}, i) = \mathsf{c}(\mathsf{r}, j)$. And, thus $\overline{\mathcal{A}_{\mathrm{PD}}(\bar{\mathsf{r}})} \simeq \overline{\mathcal{A}_{\mathrm{POS}}(\bar{\mathsf{r}})/\equiv'_c}$, where $\overline{\mathcal{A}}$ means an automaton equal to $\mathcal{A}$ but with the transition labels unmarked. Now, noting that for $\sigma \in \Sigma$, $\partial_\sigma(\mathsf{r}) = \bigcup_{i \in \mathsf{Select}(\mathsf{Pos}(\mathsf{r}), \sigma)} \overline{\partial_{\sigma_i}(\bar{\mathsf{r}})}$, the result follows.

4 Complexity of Partial Derivatives

Here we will focus on the partial derivative based automata constructions and, due to Proposition 1, on the position automaton. We will consider both the size of the automata, as well as, the complexity of the associated constructions. Moreover we restrict to standard regular expressions with union, concatenation, and Kleene star. First, we consider the $\mathcal{A}_{\mathrm{POS}}$.

Proposition 2 ([20, 32, 67]). *The position automaton $\mathcal{A}_{\mathrm{POS}}(r)$ has $|r|_\Sigma + 1$ states and the number of transitions is $\Theta(\|r\|^2)$. It can be constructed in $O(\|r\|^2)$ time and use just $O(\|r\|)$ space.*

The *star normal form* of a regular expression r, introduced by Bruggemann-Klein [20], corresponds to ensure that in any subexpression $\mathsf{s}^\star$ one has that $\varepsilon(\mathsf{s}) = \emptyset$. The conversion of an expression to star normal form can be done in linear time; the $\mathcal{A}_{\mathrm{POS}}$ of both expressions coincide; and for star normal forms its construction runs in quadratic time. Gruber and Gulan [38] extended this form to *strong star normal form* (ssnf) that is the one we will consider here.

Nicaud [63] studied the average size of $\mathcal{A}_{\mathrm{POS}}$ for the uniform distribution. Broda et al. [13], using a variant of the computation of the $\mathsf{ssnf}(\mathsf{r})$, improved the result for the number of transitions.

Proposition 3 ([63]). *Asymptotically, and as the alphabet size grows, the average number of states in $\mathcal{A}_{\mathrm{POS}}$ is $\frac{\|r\|}{2}$.*

Proposition 4 ([63, 13]). *Asymptotically, and as the alphabet size grows, the average number of transitions in $\mathcal{A}_{\mathrm{POS}}$ is $\|r\|$.*

Now, we turn to the complexity of partial derivatives and $\mathcal{A}_{\mathrm{PD}}$. The next two propositions follow directly from Equation (6).

Proposition 5 ([57],[4]Th. 3.4). *For any regular expression $r \in \mathcal{R}_k$, the following inequality holds $|\partial^+(r)| \leq |r|_\Sigma$.*

Proposition 6 ([4], Th. 3.8). *Given $r \in \mathcal{R}_k$, a partial derivative of r is either ε or a concatenation $r_1 r_2 \cdots r_n$ such that each r_i is a subexpression of r and $n-1$ is no greater than the number of occurrences of concatenations and stars in r.*

Corollary 1. *For $r_1 \in \partial^+(r)$, the size $\|r_1\|$ is $O(\|r\|^2)$.*

In general, the same bounds apply for partial derivatives by an alphabetic symbol. To improve the computation of $\partial_\sigma(\mathsf{r})$, Antimirov defined the *linear form* φ of a regular expression r that allows the computation of the partial derivatives by all alphabetic symbols in a unique transversal of the expression:

$$\varphi(\mathsf{r}) = \{\ (\sigma, \mathsf{r}') \mid \mathsf{r}' \in \partial_\sigma(\mathsf{r})\ \}. \tag{9}$$

Proposition 7 ([4, 57]). *For $r \in \mathcal{R}_k$, we have $|\varphi(r)| \leq |r|_\Sigma$ and for $(\sigma, r') \in \varphi(r)$, the size $\|r'\|$ is $O(\|r\|^2)$. If r contains no subexpression of the form $r_1^\star$, then the size $\|r'\|$ is $O(\|r\|)$.*

From the above we have

Corollary 2. *For $r \in \mathcal{R}_k$, $|\delta_{\mathrm{PD}}(r)|$ is $O(|r|_\Sigma^2)$.*

The following examples show that the above upper bounds are attained.

Example 1. Let $r_n = a_1^\star a_2^\star \cdots a_n^\star$, with $|r|_\Sigma = n, n \geq 1$. Then $\partial^+(r_n) = \{\ a_i^\star \cdots a_n^\star \mid 2 \leq i \leq n\ \}$, and $|\varphi(r_n)| = |r_n|_\Sigma = n$. The largest partial derivative has size $3n-1$, and $|\delta_{\mathrm{PD}}(r_n)| = \sum_{i=1}^{n-1} i = \frac{n(n+1)}{2}$.

Example 2. Consider $r_0 = a$ and $r_n = (r_{n-1}^{\star} a)$, for $n \geq 1$ over the unary alphabet $\{a\}$. The size of r_n is $3n + 1$, for $n \geq 0$. For $n \geq 1$, the largest partial derivative of $\partial_a(r_n) = \{\ r_i \cdots r_n \mid 1 \leq i \leq n\ \} \cup \{\varepsilon\}$ is $r_1 r_2 \cdots r_n$ whose size is

$$n - 1 + \sum_{i=1}^{n} (3i + 1) = \frac{3n^2 + 7n - 2}{2} = \Theta(n^2).$$

Although $\mathcal{A}_{\mathrm{PD}}$ is no larger than $\mathcal{A}_{\mathrm{POS}}$, the quadratic size of the partial derivatives can burden the computation of $\mathcal{A}_{\mathrm{PD}}$. Before considering the complexity of the construction algorithms, we recall some average-case estimates.

Proposition 8 ([12, 13]). *Asymptotically in the size of the expression $\mathsf{r} \in \mathcal{R}_k$, and as the alphabet size grows, the average of upper bounds of: the size of $\varphi(\mathsf{r})$ is the constant 6; the size of $\partial^+(\mathsf{r})$ is $\frac{\|\mathsf{r}\|}{4}$; the size of $\delta_{\mathrm{PD}}(\mathsf{r})$ is $\frac{\|\mathsf{r}\|}{2}$.*

In particular, we can conclude that, asymptotically, on average the size of $\mathcal{A}_{\mathrm{PD}}$ is half the size of $\mathcal{A}_{\mathrm{POS}}$. The estimation of the average size of partial derivatives is also important and was studied by Konstantinidis et al [46]. Moreover, if the regular expression is in ssnf, the size of partial derivalives are on average linear in the size of the expression. Let $\mathcal{S}_k$ be the set of regular expressions in ssnf, then, we have the following.

Proposition 9 ([46] Th.3 and Th.4). *Asymptotically and as the alphabet size grows, the average of an upper bound of the maximum size of partial derivatives of $\mathsf{r} \in \mathcal{R}_k$ is $\frac{\sqrt{\pi}}{4}(\|\mathsf{r}\|)^{\frac{3}{2}}$. For $\mathsf{s} \in \mathcal{S}_k$, that value is $\frac{\|\mathsf{s}\|}{2}$.*

Proposition 6 shows that a partial derivative is a concatenation of subexpressions of the original expression. Thus, one can estimate the average number of new concatenations when computing $\partial_\sigma(\mathsf{r})$ and $\partial^+(\mathsf{r})$.

Proposition 10 ([46, 45]). *Asymptotically and as the alphabet size grows: the average of an upper bound of the number of new concatenations in a partial derivative by an alphabetic symbol of a regular expression $\mathsf{s} \in \mathcal{S}_k$ is 14; the average of an upper bound of the number new concatenations in all partial derivatives of a regular expression $\mathsf{s} \in \mathcal{S}_k$ is $\frac{1}{8}\sqrt{\frac{\pi}{2}}\,\|\mathsf{s}\|^{\frac{3}{2}}$.*

4.1 Complexity of Building $\mathcal{A}_{\mathrm{PD}}$

Antimirov [4] presented a construction of the $\mathcal{A}_{\mathrm{PD}}$ with worst-case time complexity $O(|\mathsf{r}|_\Sigma^3 \|\mathsf{r}\|^2)$ and worst-case space complexity $O(|\mathsf{r}|_\Sigma \|\mathsf{r}\|^2)$. Mirkin's construction of $\mathcal{A}_{\mathrm{PD}}$ has a worst-case time complexity $O(\|\mathsf{r}\|^3)$. Champarnaud and Ziadi [31] presented a quadratic algorithm to construct $\mathcal{A}_{\mathrm{PD}}$ that first builds $\mathcal{A}_{\mathrm{POS}}$ and then, using Proposition 1, computes the equivalence relation $\equiv_c$ on the set of states of $\mathcal{A}_{\mathrm{POS}}$. The set of c-continuations can be computed in $O(\|\mathsf{r}\|^2 |\mathsf{r}|_\Sigma)$. To compute the relation $\equiv_c$ one can lexicographically sort the set of c-continuations using Paige and Tarjan linear algorithm [66] and then compare consecutive identical expressions. Thus, the set $\partial^+(\mathsf{r})$ can be computed in

$O(\|\mathsf{r}\|^2|\mathsf{r}|_\Sigma)$ time and space. But several improvements can be done and Champarnaud and Ziadi showed that $\mathcal{A}_{\mathrm{PD}}(\mathsf{r})$ can be computed in time and space $O(\|\mathsf{r}\|^2)$.

An improved method was proposed by Khorsi and Ziadi [43], which has worst-case time and space complexity $O(\|\mathsf{r}\||\mathsf{r}|_\Sigma)$. The main difference is the substitution of the lexicographic sorting of the c-continuations by the minimisation of an acyclic DFA and which can be performed in time $O(\|\mathsf{r}\|)$ [68]. More recently Ouardi et al [65] presented a similar algorithm using the Thompson automaton, $\mathcal{A}_{\varepsilon\text{-T}}$, instead of $\mathcal{A}_{\mathrm{POS}}$.

For practical applications, the drawbacks of these methods rely on the need to build a larger automaton and the computation of equivalence relations. Thus, it is interesting to construct the $\mathcal{A}_{\mathrm{PD}}$ in quadratic time and linear space avoiding the computation of larger automata. Using the average estimates given above, Konstantinis et al [45] presented an algorithm for computing $\mathcal{A}_{\mathrm{PD}}$ which for SSNF expressions of size n uses, on average, time $O\left(n^{3/2}\sqrt[4]{\log(n)}\right)$ and space $O\left(n^{3/2}/(\log n)^{3/4}\right)$. The regular expression and the set of its partial derivatives are represented by a directed acyclic graph (DAG) with shared common subexpressions. Flajolet et al. [35] showed that a tree of size n has, in this compact form, an expected size of $O\left(n/\sqrt{\log n}\right)$. The algorithm computes $\mathcal{A}_{\mathrm{PD}}(\mathsf{r})$ by constructing a DAG for r, and simultaneously builds the set of all partial derivatives by adding new concatenation nodes to the DAG. Empirical tests suggest that this algorithm can outperform Khorsi and Ziadi's algorithm.

A possible advantage of a direct method is its easy adaptation to other regular operations, such as intersection and shuffle [7, 16, 75], and its efficient extension to decision problems such as membership and equivalence. Partial derivatives have already been considered for solving these problems, but a fine-grain complexity analysis is needed [3, 64, 74, 59, 26].

5 Beyond Regular Languages

The notions of derivative and partial derivative are easily extended to other regularity preserving operations. We note that even for intersection and shuffle extending the position automaton is more challenging [15, 18]. However, in some cases, e.g. intersection, the Mirkin's construction may lead to automata that are not initially connected [8].

As left-quotients are defined for any formal language, derivatives have been defined for context-free languages both considering grammars and μ-expressions [1, 76]. Interestingly, parsing context-free grammars with derivatives can be achieved in cubic time. Parsing expression grammars (PEG) are a recognition-based formalism for which parsing can be achieved in linear time. Deterministic context-free languages are recognisable by PEGs but it is not known if all context-free grammars are recognisable, although some non-context-free languages are. Derivatives for PEGs were proposed in [60].

Derivatives of weighted rational expressions, that represent formal power series with coefficients in a semiring, have also been extensively studied [50, 29,

70, 71]. In this case, derivatives are also connected with the notion of quotient of a series. In the same manner, if one consider trees instead of words the above methods can be extended to tree regular expressions and tree automata [55, 56, 54].

More recently, partial derivatives for regular expressions with labels over finitely generated monoids (possible non-free) were studied by Konstantinidis et al [47] and also by Demaille [34]. In particular, those expressions allow to represent (weighted) rational relations. In this case, using an appropriate version of the linear form of an expression, the equations (8) and (6) hold and a partial derivative automaton can be defined. Lombardy and Sakarovitch [51] expanded and generalised this approach, showing that the partial derivative automaton is a quotient of position automaton, even considering weighted regular expressions over non free monoids.

Finally we briefly point to the fast literature on category theory of automata and algebraic and coalgeraic approaches to characterise general state-based systems where the notion of derivative plays also an important role [69, 73, 10, 42].

References

1. Adams, M.D., Hollenbeck, C., Might, M.: On the complexity and performance of parsing with derivatives. In: Krintz, C., Berger, E. (eds.) Proc. 37th ACM SIGPLAN PLDI. pp. 224–236. ACM (2016). https://doi.org/10.1145/2908080.2908128
2. Almeida, M., Moreira, N., Reis, R.: Antimirov and Mosses' rewrite system revisited. Int. J. Found. Comput. S. **20**(4), 669–684 (August 2009). https://doi.org/10.1142/S0129054109006802
3. Almeida, M., Moreira, N., Reis, R.: Testing equivalence of regular languages. J. Autom. Lang. Comb. **15**(1/2), 7–25 (2010). https://doi.org/10.25596/jalc-2010-007
4. Antimirov, V.M.: Partial derivatives of regular expressions and finite automaton constructions. Theoret. Comput. Sci. **155**(2), 291–319 (1996). https://doi.org/http://dx.doi.org/10.1016/0304-3975(95)00182-4
5. Antimirov, V.M., Mosses, P.: Rewriting extended regular expressions. In: Rozenberg, G., Salomaa, A. (eds.) Developments in Language Theory. pp. 195 – 209. World Scientific (1994)
6. Asperti, A., Coen, C.S., Tassi, E.: Regular expressions, au point. CoRR **abs/1010.2604** (2010), `http://arxiv.org/abs/1010.2604`
7. Bastos, R., Broda, S., Machiavelo, A., Moreira, N., Reis, R.: On the state complexity of partial derivative automata for regular expressions with intersection. In: Câmpeanu, C., Manea, F., Shallit, J. (eds.) Proc. 18th DCFS 2016. LNCS, vol. 9777, pp. 45–59. Springer (2016). https://doi.org/10.1007/978-3-319-41114-9
8. Bastos, R., Broda, S., Machiavelo, A., Moreira, N., Reis, R.: On the average complexity of partial derivative automata for semi-extended expressions. J. Autom. Lang. Comb. **22**(1–3), 5–28 (2017). https://doi.org/10.25596/jalc-2017-005
9. Berry, G., Sethi, R.: From regular expressions to deterministic automata. Theoret. Comput. Sci. **48**, 117–126 (1986)
10. Bonchi, F., Pous, D.: Checking NFA equivalence with bisimulations up to congruence. In: Giacobazzi, R., Cousot, R. (eds.) Proc. 40th POPL '13. pp. 457–468. ACM (2013). https://doi.org/10.1145/2429069.2429124

11. Broda, S., Holzer, M., Maia, E., Moreira, N., Reis, R.: A mesh of automata. Inf. Comput. **265**, 94–111 (2019). https://doi.org/10.1016/j.ic.2019.01.003
12. Broda, S., Machiavelo, A., Moreira, N., Reis, R.: On the average state complexity of partial derivative automata: an analytic combinatorics approach. Int. J. Found. Comput. Sci. **22**(7), 1593–1606 (2011). https://doi.org/10.1142/S012905
13. Broda, S., Machiavelo, A., Moreira, N., Reis, R.: On the average size of Glushkov and partial derivative automata. Int. J. Found. Comput. S. **23**(5), 969–984 (2012). https://doi.org/10.1142/S0129054112400400
14. Broda, S., Machiavelo, A., Moreira, N., Reis, R.: Average size of automata constructions from regular expressions. Bulletin of the European Association for Theoretical Computer Science **116**, 167–192 (June 2015), `http://bulletin.eatcs.org/index.php/beatcs/article/view/352/334`
15. Broda, S., Machiavelo, A., Moreira, N., Reis, R.: Position automaton construction for regular expressions with intersection. In: Brlek, S., Reutenauer, C. (eds.) Proc. 20th DLT 2016. LNCS, vol. 9840, pp. 51–63. Springer (2016). https://doi.org/10.1007/978-3-662-53131-7 5
16. Broda, S., Machiavelo, A., Moreira, N., Reis, R.: Automata for regular expressions with shuffle. Inf. Comput. **259**(2), 162–173 (2018). https://doi.org/10.1016/j.ic.2017.08.013
17. Broda, S., Machiavelo, A., Moreira, N., Reis, R.: Analytic combinatorics and descriptional complexity of regular languages on average. ACM SIGACT News **51**(1), 38–56 (March 2020)
18. Broda, S., Machiavelo, A., Moreira, N., Reis, R.: Location based automata for expressions with shuffle. In: Leporati, A., Martín-Vide, C., Shapira, D., Zandron, C. (eds.) Proc. 15th LATA 2021. vol. 12638, pp. 43–54. Springer (2021). https://doi.org/10.1007/978-3-030-68195-1_4
19. Broda, S., Maia, E., Moreira, N., Reis, R.: The prefix automaton. J. Autom. Lang. Comb. **26**(1-2), 17–53 (2021). https://doi.org/10.25596/jalc-2021-017
20. Brüggemann-Klein, A.: Regular expressions into finite automata. Theoret. Comput. Sci. **48**, 197–213 (1993)
21. Brzozowski, J.A., Jr., E.J.M.: Signal flow graph techniques for sequential circuit state diagrams. IEEE Trans. on Electronic Computers **EC-12**(2), 67–76 (1963)
22. Brzozowski, J.: Regular expression techniques for sequential circuits. Ph.D. thesis, Department of Electrical Engineering, Princeton University (1962)
23. Brzozowski, J.A.: Quotient complexity of regular languages **15**(1/2), 71–89 (2010). https://doi.org/10.25596/jalc-2010-071
24. Brzozowski, J.A., Leiss, E.L.: On equations for regular languages, finite automata, and sequential networks. Theor. Comput. Sci. **10**, 19–35 (1980). https://doi.org/10.1016/0304-3975(80)90069-9
25. Brzozowski, J.: Derivatives of regular expressions. J. ACM **11**(4), 481–494 (1964). https://doi.org/10.1145/321239.321249
26. Cardoso, E.M., Amaro, M., da Silva Feitosa, S., dos Santos Reis, L.V., Bois, A.R.D., Ribeiro, R.G.: The design of a verified derivative-based parsing tool for regular expressions. CLEI Electron. J. **24**(3) (2021). https://doi.org/10.19153/cleiej.24.3.2
27. Champarnaud, J.M., Ziadi, D.: From Mirkin's prebases to Antimirov's word partial derivatives. Fundam. Inform. **45**(3), 195–205 (2001)
28. Champarnaud, J.M., Ziadi, D.: Canonical derivatives, partial derivatives and finite automaton constructions. Theoret. Comput. Sci. **289**, 137–163 (2002). https://doi.org/10.1016/S0304-3975(01)00267-5

29. Champarnaud, J., Ouardi, F., Ziadi, D.: An efficient computation of the equation K-automaton of a regular K-expression. In: Harju, T., Karhumäki, J., Lepistö, A. (eds.) Proc. 11th DLT 2007. LNCS, vol. 4588, pp. 145–156. Springer (2007). https://doi.org/10.1007/978-3-540-73208-2_16
30. Champarnaud, J., Ponty, J., Ziadi, D.: From regular expressions to finite automata. Int. J. Comput. Math. **72**(4), 415–431 (1999). https://doi.org/10.1080/00207169908804865
31. Champarnaud, J., Ziadi, D.: From c-continuations to new quadratic algorithms for automaton synthesis. Intern. Journ. of Alg. and Comp. **11**(6), 707–736 (2001)
32. Chang, C., Paige, R.: From regular expressions to dfa's using compressed nfa's. Theor. Comput. Sci. **178**(1-2), 1–36 (1997). https://doi.org/10.1016/S0304-3975(96)00140-5
33. Conway, J.H.: Regular Algebra and Finite Machines. Chapman and Hall (1971)
34. Demaille, A.: Derived-term automata of multitape rational expressions. In: Han, Y., Salomaa, K. (eds.) Proc. 21st CIAA. LNCS, vol. 9705, pp. 51–63. Springer (2016). https://doi.org/10.1007/978-3-319-40946-7_5
35. Flajolet, P., Sipala, P., Steyaert, J.: Analytic variations on the common subexpression problem. In: Paterson, M. (ed.) Proc. 17th ICALP 90. vol. 443, pp. 220–234. Springer (1990). https://doi.org/10.1007/BFb0032034
36. Ginzburg, A.: A procedure for checking equality of regular expressions. J. ACM **14**(2), 355–362 (1967). https://doi.org/10.1145/321386.321399
37. Glushkov, V.M.: The abstract theory of automata. Russ. Math. Surv. **16**, 1–53 (1961)
38. Gruber, H., Gulan, S.: Simplifying regular expressions. In: Dediu, A., Fernau, H., Martín-Vide, C. (eds.) Proc. 4th LATA. vol. 6031, pp. 285–296 (2010). https://doi.org/10.1007/978-3-642-13089-2_24
39. Gruber, H., Holzer, M.: From finite automata to regular expressions and back—a summary on descriptional complexity. Int. J. Found. Comput. S. **26**(8), 1009–1040 (Dec 2015). https://doi.org/10.1142/S0129054115400110
40. Holzer, M., Kutrib, M.: The complexity of regular(-like) expressions. Int. J. Found. Comput. Sci. **22**(7), 1533–1548 (2011). https://doi.org/10.1142/S0129054111008866, `https://doi.org/10.1142/S0129054111008866`
41. Hopcroft, J., Karp, R.M.: A linear algorithm for testing equivalence of finite automata. Tech. Rep. TR 71 -114, University of California, Berkeley, California (1971)
42. Kappé, T., Brunet, P., Luttik, B., Silva, A., Zanasi, F.: Brzozowski goes concurrent - A Kleene theorem for pomset languages. In: Meyer, R., Nestmann, U. (eds.) Proc. 28th CONCUR 2017. LIPIcs, vol. 85, pp. 25:1–25:16. Schloss Dagstuhl - Leibniz-Zentrum für Informatik (2017). https://doi.org/10.4230/LIPIcs.CONCUR.2017.25
43. Khorsi, A., Ouardi, F., Ziadi, D.: Fast equation automaton computation. J. Discrete Algorithms **6**(3), 433–448 (2008). https://doi.org/10.1016/j.jda.2007.10.003
44. Kleene, S.C.: Representation of events in nerve nets and finite automata. In: Shannon, C.E., McCarthy, J. (eds.) Automata Studies, pp. 3–41. Princeton University Press, Princeton (1956)
45. Konstantinidis, S., Machiavelo, A., Moreira, N., Reis, R.: Partial derivative automaton by compressing regular expressions. In: Yan, Y., Ko, S. (eds.) Proc. 23rd DCFS 2021. LNCS, vol. 13037, pp. 100–112. Springer (2021). https://doi.org/https://doi.org/10.1007/978-3-030-93489-7
46. Konstantinidis, S., Moreira, N., Reis, R.: On the size of partial derivatives and the word membership problem. Acta Informatica **58**(4), 357–375 (2021). https://doi.org/10.1007/s00236-021-00399-6

47. Konstantinidis, S., Moreira, N., Reis, R.: Partial derivatives of regular expressions over alphabet-invariant and user-defined labels. Theor. Comput. Sci. **870**, 103–120 (2021). https://doi.org/10.1016/j.tcs.2020.12.029
48. Krob, D.: Differentiation of k-rational expressions. Int. J. Algebra Comput. **2**(1), 57–88 (1992). https://doi.org/10.1142/S0218196792000062
49. Leiss, E.L.: The complexity of restricted regular expressions and the synthesis problem for finite automata. J. Comput. Syst. Sci. **23**(3), 348–354 (1981). https://doi.org/10.1016/0022-0000(81)90070-2
50. Lombardy, S., Sakarovitch, J.: Derivatives of rational expressions with multiplicity. Theor. Comput. Sci. **332**(1-3), 141–177 (2005). https://doi.org/10.1016/j.tcs.2004.10.016
51. Lombardy, S., Sakarovitch, J.: Derived terms without derivation a shifted perspective on the derived-term automaton. Journal of Computer Science and Cybernetics **37**(3), 201–221 (oct 2021). https://doi.org/10.15625/1813-9663/37/3/16263
52. McCulloch, W., Pitts, W.: A logical calculus of the ideas immanent in nervous activity. Bulletin of Mathematical Biophisics (5), 115–133 (1943)
53. McNaughton, R., Yamada, H.: Regular expressions and state graphs for automata. IEEE Transactions on Electronic Computers **9**, 39–47 (1960)
54. Mignot, L.: A unified implementation of automata and expression structures, and of the associated algorithms using enriched categories. CoRR **abs/2012.10641** (2020), `https://arxiv.org/abs/2012.10641`
55. Mignot, L., Sebti, N.O., Ziadi, D.: Tree automata constructions from regular expressions: a comparative study. Fundam. Informaticae **156**(1), 69–94 (2017). https://doi.org/10.3233/FI-2017-1598
56. Mignot, L., Sebti, N.O., Ziadi, D.: An efficient algorithm for the construction of the equation tree automaton. Int. J. Found. Comput. Sci. **29**(6), 951–978 (2018). https://doi.org/10.1142/S0129054118500156
57. Mirkin, B.G.: An algorithm for constructing a base in a language of regular expressions. Engineering Cybernetics **5**, 51—57 (1966)
58. Mizoguchi, Y., Ohtsuka, H., Kawahara, Y.: Symbolic calculus of regular expressions. Bulletin of Informatics and Cybernetics **22**(3-4), 165–170 (1987)
59. Moreira, N., Pereira, D., de Sousa, S.M.: Deciding Kleene algebra terms (in)equivalence in Coq. Journal of Logical and Algebraic Methods in Programming **84**(3), 377–401 (2015). https://doi.org/doi:10.1016/j.jlamp.2014.12.004
60. Moss, A.: Simplified parsing expression derivatives. In: Leporati, A., Martín-Vide, C., Shapira, D., Zandron, C. (eds.) Proc. 14th LATA 2020. LNCS, vol. 12038, pp. 425–436. Springer (2020). https://doi.org/10.1007/978-3-030-40608-0_30
61. Myhill, J.: Finite automata and representation of events. In: Fundamental Concepts in the Theory of Systems, vol. 57. Wright Air Development Center (1957)
62. Nerode, A.: Linear automaton transformations. Procedings of the American Mathematical Society pp. 541–544 (1958)
63. Nicaud, C.: On the average size of Glushkov's automata. In: Dediu, A., Ionescu, A.M., Vide, C.M. (eds.) Proc. 3rd LATA. vol. 5457, pp. 626–637 (2009). https://doi.org/10.1007/978-3-642-00982-2_53
64. Nipkow, T., Traytel, D.: Unified decision procedures for regular expression equivalence. In: Klein, G., Gamboa, R. (eds.) 5th ITP. LNCS, vol. 8558, pp. 450–466. Springer (2014). https://doi.org/10.1007/978-3-319-08970-6-29
65. Ouardi, F., Lotfi, Z., Elghadyry, B.: Efficient construction of the equation automaton. Algorithms **14**(8), 238 (2021). https://doi.org/10.3390/a14080238
66. Paige, R., Tarjan, R.E.: Three partition refinement algorithms. SIAM J. Comput. **16**(6), 973–989 (1987). https://doi.org/http://dx.doi.org/10.1137/0216062

67. Ponty, J., Ziadi, D., Champarnaud, J.: A new quadratic algorithm to convert a regular expression into an automaton. In: Raymond, D.R., Wood, D., Yu, S. (eds.) Proc. WIA'96. LNCS, vol. 1260, pp. 109–119. Springer (1996). https://doi.org/10.1007/3-540-63174-7_9
68. Revuz, D.: Minimisation of acyclic deterministic automata in linear time. Theoret. Comput. Sci. **92**(1), 181–189 (1992)
69. Rutten, J.: Behavioural differential equations: a coinductive calculus of streams, automata, and power series. Theoret. Comput. Sci. **208**(1–3), 1–53 (2003). https://doi.org/10.1016/S0304-3975(02)00895-2
70. Sakarovitch, J.: Elements of Automata Theory. Cambridge University Press (2009). https://doi.org/10.1017/CBO9781139195218
71. Sakarovitch, J.: Automata and rational expressions. In: Pin, J. (ed.) Handbook of Automata Theory, pp. 39–78. European Mathematical Society Publishing House, Zürich, Switzerland (2021). https://doi.org/10.4171/Automata-1/2
72. Salomaa, A.: Two complete axiom systems for the algebra of regular events. Journal of the Association for Computing Machinery **13**(1), 158–169 (1966)
73. Silva, A., Bonchi, F., Bonsangue, M.M., Rutten, J.J.M.M.: Quantitative Kleene coalgebras. Inf. Comput. **209**(5), 822–849 (2011). https://doi.org/10.1016/j.ic.2010.09.007
74. Sulzmann, M., Lu, K.Z.M.: Regular expression sub-matching using partial derivatives. In: Schreye, D.D., Janssens, G., King, A. (eds.) Proc. PPDP'12. pp. 79–90. ACM (2012). https://doi.org/10.1145/2370776.2370788
75. Sulzmann, M., Thiemann, P.: Derivatives and partial derivatives for regular shuffle expressions. J. Comput. Syst. Sci. **104**, 323–341 (2019). https://doi.org/10.1016/j.jcss.2016.11.010
76. Thiemann, P.: Partial derivatives for context-free languages: From μ-regular expressions to pushdown automata. In: Esparza, J., Murawski, A.S. (eds.) Proc. 20th FOSSACS. vol. 10203, pp. 248–264 (2017). https://doi.org/10.1007/978-3-662-54458-7_15
77. Thompson, K.: Regular expression search algorithm. Commun. ACM **11**(6), 410–422 (1968)
78. Yamamoto, H.: A new finite automaton construction for regular expressions. In: Bensch, S., Freund, R., Otto, F. (eds.) 6th NCMA. books@ocg.at, vol. 304, pp. 249–264. Österreichische Computer Gesellschaft (2014)