# Partial Derivatives of an Extended Regular Expression

Pascal Caron, Jean-Marc Champarnaud, and Ludovic Mignot

LITIS, Université de Rouen, 76801 Saint-Étienne du Rouvray Cedex, France
{pascal.caron,jean-marc.champarnaud,ludovic.mignot}@univ-rouen.fr

**Abstract.** The notion of expression derivative due to Brzozowski leads to the construction of a deterministic automaton from an extended regular expression, whereas the notion of partial derivative due to Antimirov leads to the construction of a non-deterministic automaton from a simple regular expression. In this paper, we generalize Antimirov partial derivatives to regular expressions extended to complementation and intersection. For a simple regular expression with $n$ symbols, Antimirov automaton has at most $n+1$ states. As far as an extended regular expression is concerned, we show that the number of states can be exponential.

## 1 Introduction

Regular expressions are a basic tool for describing patterns in a text. This is the reason why they are used in numerous domains that involve pattern specification or pattern matching, such as electronical document processing, bio-informatics or data bases. An additional advantage of regular expressions is that they can be transformed into an equivalent machine, called a finite automaton, that makes it possible to automatically decide whether a word belongs to the language denoted by an expression or not.

Simple regular expressions only contain sum, concatenation product and Kleene star operators whereas extended regular expressions in addition contain boolean operators such as complementation or intersection. Simple regular expressions have been extensively investigated. Numerous algorithms have been designed in particular for converting a regular expression into a finite automaton. These algorithms can be partitioned into two main categories. The algorithms of the first category are based on the notion of position (of a symbol occurrence in the expression). It is the case of the algorithm due to Glushkov [9] and to McNaughton and Yamada [12] that computes a non-deterministic automaton with $n + 1$ states from a $n$-symbol occurrence expression. Let us notice that, under some assumptions, the inductive algorithm [11,4] computes the same automaton. The algorithms of the second category are based on the computation of expression derivatives that is similar to the computation of language quotients [13,14]. It is the case of the algorithm of expression derivatives due to Brzozowski [3] that computes a deterministic automaton and of the algorithm of partial derivatives due to Antimirov [1] that computes a non-deterministic automaton with

A.-H. Dediu, S. Inenaga, and C. Martín-Vide (Eds.): LATA 2011, LNCS 6638, pp. 179–191, 2011.

at most $n + 1$ states if there are $n$ symbol occurrences in the expression. Let us remark that the relations that exist between the two notions of position and of expression derivative have been studied in [2,5].

As far as extended regular expressions are concerned, many complexity studies [6] have been realized; we will consider in the following the results of Gelade and Neven [8] about the succinctness of the operations of complementation and intersection. On the opposite, there exist few algorithms performing the conversion of an extended regular expression into an automaton. Actually, boolean operators (except for the sum) are not compatible with the notion of position and thus Glushkov algorithm cannot be extended from simple to extended regular expressions. The inductive algorithm still works for extended regular expressions, but its performance is penalized by the determinization steps required by the automaton-implementation of some boolean operations. Actually, extended regular expressions are handled by the algorithm of Brzozowski [3]; however, a deterministic automaton is computed, which can be a drawback regarding to space complexity. As far as partial derivatives are concerned, here is what Antimirov reports in the conclusion of his paper [1]: "It would be useful to find an appropriate definition of partial derivatives of *extended* regular expressions (with intersection, complementation, and other operations). Then, in particular, our NFA construction would directly extend to this class of regular expressions."

In this paper, we extend the computation of partial derivatives in order to handle the operations of complementation and intersection. It allows us to generalize the two automaton constructions designed by Antimirov: the non-deterministic derivated term automaton and the deterministic partial derivative automaton. We also show that the partial derivative automaton may have a number of states exponential with respect to the size of the expression.

The main notions used in this paper as well as the computation of expression derivatives and partial derivatives are recalled in the next section. A generalization of the computation of partial derivatives to extended regular expressions is introduced in Section 3. Section 4 and Section 5 are respectively devoted to the construction of the derivated term automaton and of the partial derivative automaton that both recognize the language denoted by a given extended regular expression.

## 2 Preliminaries

A *finite automaton* $A$ is a 5-tuple $(\Sigma, Q, I, F, \delta)$ with $\Sigma$ the *alphabet* (a finite set of symbols), $Q$ a finite set of *states*, $I \subset Q$ the set of *initial states*, $F \subset Q$ the set of *final states* and $\delta \subset Q \times \Sigma \times Q$ the set of *transitions*. The set $\delta$ is equivalent to the function from $Q \times \Sigma$ to $2^Q$ defined by: $q' \in \delta(q, a)$ if and only if $(q, a, q') \in \delta$. The domain of the function $\delta$ is extended to $2^Q \times \Sigma^*$ as follows: $\forall P \subset Q$, $\delta(P, \varepsilon) = P$, $\delta(P, a) = \bigcup_{p \in P} \delta(p, a)$ and $\delta(P, a \cdot w) = \delta(\delta(P, a), w)$. The automaton $A$ *recognizes* the language $L(A) = \{w \in \Sigma^* \mid \delta(I, w) \cap F \neq \emptyset\}$. The automaton $A$ is *deterministic* if $\#I = 1 \ \wedge \ \forall (q, a) \in Q \times \Sigma$, $\#\delta(q, a) \leq 1$. A deterministic automaton is *complete* if $\forall (q, a) \in Q \times \Sigma$, $\#\delta(q, a) = 1$.

For every automaton $A$, there exists a complete deterministic automaton $A'$ such that $L(A') = L(A)$ [15].

Let f be a boolean operator of arity $k$. An *extended regular expression* $E$ over an alphabet $\Sigma$ is inductively defined by $E = \emptyset$, $E = \varepsilon$, $E = a$, $E = \mathrm{f}(E_1, \ldots, E_k)$, $E = (F \cdot G)$, $E = (F^*)$ where $a \in \Sigma$ and $F$, $G$, $E_1, \ldots, E_k$ are extended regular expressions. The regular expression $E$ is said to be a *simple expression* if it only contains sum, concatenation product and Kleene star operators.

The *language denoted by* the expression $E$ is inductively defined by

$$L(\emptyset) = \emptyset,\ L(\varepsilon) = \{\varepsilon\},\ L(a) = \{a\}\ \forall a \in \Sigma,\ L(F \cdot G) = L(F) \cdot L(G),$$
$$L(F^*) = L(F)^*,\ L(\mathrm{f}(E_1, \ldots, E_k)) = \mathrm{f}_L(L(E_1), \ldots, L(E_k)),$$

where $\mathrm{f}_L$ is the language operator associated with the operator[1] f.

A language $L$ is *regular* if and only if there exists a simple regular expression $E$ such that $L(E) = L$. It has been proved by Kleene [10] that a language is regular if and only if it is recognized by a finite automaton. Moreover, given a complete deterministic automaton $A$, a deterministic automaton $A'$ such that $L(A') = \neg L(A)$ can be constructed by setting non final the final states and vice versa. Therefore, the set of regular languages is closed under any boolean operator.

The *quotient of a language* $L$ w.r.t. a symbol $a$ is the language $a^{-1}(L) = \{w \in \Sigma^* \mid aw \in L\}$. It can be recursively computed as follows:

$$a^{-1}(\emptyset) = a^{-1}(\{\varepsilon\}) = \emptyset,\ a^{-1}(\{b\}) = \{\varepsilon\} \text{ if } a = b,\ \emptyset \text{ otherwise},$$
$$a^{-1}(L_1 \cup L_2) = a^{-1}(L_1) \cup a^{-1}(L_2),\ a^{-1}(L_1^*) = a^{-1}(L_1) \cdot L_1^*$$
$$a^{-1}(L_1 \cdot L_2) = \begin{cases} a^{-1}(L_1) \cdot L_2 \cup a^{-1}(L_2) & \text{if } \varepsilon \in L_1, \\ a^{-1}(L_1) \cdot L_2 & \text{otherwise,} \end{cases}$$
$$a^{-1}(\mathrm{f}_L(L_1, \ldots, L_k)) = \mathrm{f}_L(a^{-1}(L_1), \ldots, a^{-1}(L_k)).$$

The quotient $w^{-1}(L)$ of $L$ w.r.t. the word $w \in \Sigma^*$ is the language $\{w' \in \Sigma^* \mid w \cdot w' \in L\}$. It can be recursively computed as follows: $\varepsilon^{-1}(L) = L$, $(aw')^{-1}(L) = w'^{-1}(a^{-1}(L))$ with $a \in \Sigma$ and $w' \in \Sigma^*$. Myhill-Nerode theorem [13,14] states that a language $L$ is regular if and only if the set of quotients $\{u^{-1}(L) \mid u \in \Sigma^*\}$ is finite.

The notion of derivative of an expression has been introduced by Brzozowski [3]. Let $E$ be an extended regular expression over an alphabet $\Sigma$ and let $a$ and $b$ be two distinct symbols of $\Sigma$. The *derivative of* $E$ w.r.t. $a$ is the expression $\frac{d}{d_a}(E)$ inductively computed as follows[2]:

$$\tfrac{d}{d_a}(\emptyset) = \tfrac{d}{d_a}(\varepsilon) = \tfrac{d}{d_a}(b) = \emptyset,\ \tfrac{d}{d_a}(a) = \varepsilon,$$
$$\tfrac{d}{d_a}(\mathrm{f}(E_1, \ldots, E_k)) = \mathrm{f}(\tfrac{d}{d_a}(E_1), \ldots, \tfrac{d}{d_a}(E_k)),\ \tfrac{d}{d_a}(F^*) = \tfrac{d}{d_a}(F) \cdot F^*,$$
$$\tfrac{d}{d_a}(F \cdot G) = \begin{cases} \frac{d}{d_a}(F) \cdot G + \frac{d}{d_a}(G) & \text{if } \varepsilon \in L(F), \\ \frac{d}{d_a}(F) \cdot G & \text{otherwise.} \end{cases}$$

The derivative of $E$ w.r.t. a word $w$ of $\Sigma^*$ is defined by:

$$\tfrac{d}{d_w}(E) = \begin{cases} \frac{d}{d_{w'}}(\frac{d}{d_a}(E)) & \text{if } w = a \cdot w' \text{ with } a \in \Sigma \text{ and } w' \in \Sigma^* \\ E & \text{if } w = \varepsilon. \end{cases}$$

[1] For instance $L(\neg E) = \neg L(E)$, $L(E + F) = L(E) \cup L(F)$.

[2] This notation is used in order to distinguish the derivative of an expression from the quotient of a language.

The set of derivatives of an expression $E$ is not necessarily finite. It has been proved by Brzozowski [3] that it is sufficient to use the ACI equivalence (that is based on the associativity, the commutativity and the idempotence of the sum) to obtain a finite set of derivatives: the set $\mathcal{D}_E$ of *dissimilar derivatives*. Given a class of ACI-equivalent expressions, a unique representative can be obtained after deleting parenthesis (associativity), ordering terms of each sum (commutativity) and deleting redundant subexpressions (idempotence). Let $E_{\sim_s}$ be the unique representative of the class of the expression $E$. The set of dissimilar derivatives can be computed as follows:

$$\frac{d'}{d'_a}(\emptyset) = \frac{d'}{d'_a}(\varepsilon) = \frac{d'}{d'_a}(b) = \emptyset,\ \frac{d'}{d'_a}(a) = \varepsilon,$$
$$\frac{d'}{d'_a}(\mathrm{f}(E_1, \ldots, E_k)) = (\mathrm{f}(\frac{d'}{d'_a}(E_1), \ldots, \frac{d'}{d'_a}(E_k)))_{\sim_s},$$
$$\frac{d'}{d'_a}(F^*) = \frac{d'}{d'_a}(F) \cdot F^*,$$
$$\frac{d'}{d'_a}(F \cdot G) = \begin{cases} (\frac{d'}{d'_a}(F) \cdot G + \frac{d'}{d'_a}(G))_{\sim_s} & \text{if } \varepsilon \in L(F), \\ (\frac{d'}{d'_a}(F) \cdot G)_{\sim_s} & \text{otherwise.} \end{cases}$$

*Example 1.* Let us consider the expression $E = a^* \cdot a^*$ over the alphabet $\Sigma = \{a\}$. The set of derivatives of $E$ is infinite since for every $w \in \Sigma^*$, $\frac{d}{d_{wa}}(E) = \frac{d}{d_w}(E) + a^*$. On the opposite, since $\frac{d}{d_{aa}}(E) = a^* \cdot a^* + a^* + a^* \sim_s a^* \cdot a^* + a^* = \frac{d}{d_a}(E)$, it holds $\frac{d'}{d'_{aa}}(E) = \frac{d'}{d'_a}(E)$. Thus the set of dissimilar derivatives of $E$ is $\mathcal{D}_E = \{a^*a^*, a^*a^* + a^*\}$.

The *derivative automaton* $B = (\Sigma, Q, q_0, F, \delta)$ of an extended regular expression $E$ over an alphabet $\Sigma$ is defined by $Q = \mathcal{D}_E$, $q_0 = E$, $F = \{q \in Q \mid \varepsilon \in L(q)\}$, $\delta = \{(q, a, q') \in Q \times \Sigma \times Q \mid \frac{d'}{d'_a}(q) = q'\}$. The automaton $B$ is deterministic and it recognizes the language $L(E)$. Its size can be exponentially larger than the number of symbols of $E$ (see Example 2).

*Example 2.* Consider the expression $E = (a+b)^*a(a+b)^n$. The set of its derivatives can be computed as follows, where $\Sigma = a + b$:

$\frac{d'}{d'_a}(E) = E + \Sigma^n$ $\quad \frac{d'}{d'_{aa}}(E) = E + \Sigma^n + \Sigma^{n-1}$ $\quad \frac{d'}{d'_a}(\Sigma^k) = \frac{d'}{d'_b}(\Sigma^k) = \Sigma^{k-1}$
$\frac{d'}{d'_b}(E) = E$ $\quad \frac{d'}{d'_{ab}}(E) = E + \Sigma^{n-1}$

The set $\mathcal{D}_E$ of dissimilar derivatives of $E$ is equal to the set $E \cup \{E + \sum_{F \in \mathcal{F}} F \mid \mathcal{F} \subset \{\Sigma^n, \ldots, \Sigma, \varepsilon\}\}$. Consequently, $\#\mathcal{D}_E = 1 + 2^{n+1}$. The derivative automaton of $E$ is presented in Figure 1.

Antimirov algorithm [1] constructs a non-deterministic automaton from a simple regular expression $E$. It is based on the *partial derivative* computation. The partial derivative of a simple regular expression $E$ w.r.t. a symbol $a$ is the set $\frac{\partial}{\partial_a}(E)$ of expressions defined as follows:

$$\frac{\partial}{\partial_a}(\emptyset) = \frac{\partial}{\partial_a}(\varepsilon) = \frac{\partial}{\partial_a}(b) = \emptyset,\ \frac{\partial}{\partial_a}(a) = \{\varepsilon\},$$
$$\frac{\partial}{\partial_a}(F + G) = \frac{\partial}{\partial_a}(F) \cup \frac{\partial}{\partial_a}(G),\ \frac{\partial}{\partial_a}(F^*) = \frac{\partial}{\partial_a}(F) \cdot F^*,$$
$$\frac{\partial}{\partial_a}(F \cdot G) = \begin{cases} \frac{\partial}{\partial_a}(F) \cdot G \cup \frac{\partial}{\partial_a}(G) & \text{if } \varepsilon \in L(F), \\ \frac{\partial}{\partial_a}(F) \cdot G & \text{otherwise,} \end{cases}$$

with for a set $\mathcal{E}$ of expressions, $\mathcal{E} \cdot F = \bigcup_{E \in \mathcal{E}} E \cdot F$.

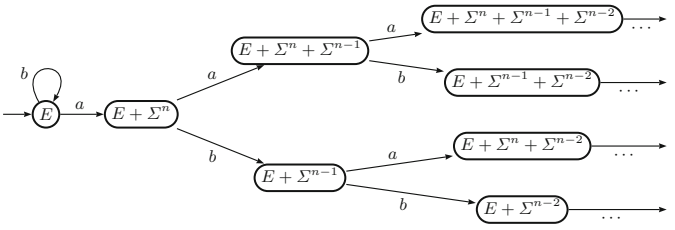


**Fig. 1.** The derivative automaton of $E = (a + b)^* a(a + b)^n$

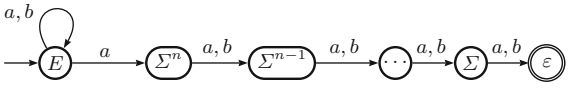


**Fig. 2.** The derivated term automaton of $E = (a + b)^* a(a + b)^n$

The partial derivative of $E$ w.r.t. a word $w$ of $\Sigma^*$ is computed as follows:

$$\frac{\partial}{\partial_w}(E) = \begin{cases} \frac{\partial}{\partial_{w'}}(\frac{\partial}{\partial_a}(E)) & \text{if } w = a \cdot w' \text{ with } a \in \Sigma \text{ and } w' \in \Sigma^*, \\ \frac{\partial}{\partial w}(E) = \{E\} & \text{if } w = \varepsilon, \end{cases}$$

with for a set $\mathcal{E}$ of expressions, $\frac{\partial}{\partial_a}(\mathcal{E}) = \bigcup_{E \in \mathcal{E}} \frac{\partial}{\partial_a}(E)$.

Every element of a partial derivative is called a *derivated term* of $E$. It has been shown by Antimirov [1] that the set $\mathcal{D}'_E$ of the derivated terms of $E$ is such that $\#\mathcal{D}'_E \leq n + 1$. The *derivated term automaton* $A = (\Sigma, Q, q_0, T, \delta)$ of a simple regular expression $E$ is defined as follows: $Q = \mathcal{D}'_E$, $q_0 = E$, $F = \{q \in Q \mid \varepsilon \in L(q)\}$, $\delta = \{(q, a, q') \in Q \times \Sigma \times Q \mid q' \in \frac{\partial}{\partial_a}(q)\}$. The automaton $A$ recognizes the language $L(E)$.

*Example 3.* Consider the expression $E = (a + b)^* a(a + b)^n$ of Example 2. The derivated terms of $E$ are computed as follows:

$$\begin{aligned} \frac{\partial}{\partial_a}((a + b)^* a(a + b)^n) &= \frac{\partial}{\partial_a}((a + b)^*) \cdot a(a + b)^n \cup \frac{\partial}{\partial_a}(a(a + b)^n) \\ &= \{E\} \cup \{\Sigma^n\}, \\ \frac{\partial}{\partial_b}(E) = \{E\}, \quad \frac{\partial}{\partial_a}(\Sigma^k) &= \frac{\partial}{\partial_b}(\Sigma^k) = \{\Sigma^{k-1}\}. \end{aligned}$$

The set of derivated terms of $E$ is $\mathcal{D}'_E = \{E, \Sigma^n, \Sigma^{n-1}, \ldots, \varepsilon\}$; the number of derivated terms is equal to $n + 2$. The derivated term automaton of $E$ is represented in Figure 2.

## 3 Partial Derivatives of an Extended Expression

This section describes two extensions of the computation of partial derivatives for extended regular expressions. The first one is a natural extension in which the simple operators ($+$, $\cdot$ and $^*$) activate the standard partial derivative computation, whereas the boolean operators (except for the sum) activate the Brzozowski derivative computation. The interest of this technique is to make easier the

understanding of the second extension, where partial derivatives of a new type are computed.

### 3.1 A Natural Extension

A natural extension of the partial derivative computation is achieved by assuming that the partial derivative of a boolean operator (except for the sum) is the singleton equal to the derivative of this operator.

**Definition 1.** *The partial derivative of an extended regular expression w.r.t. a symbol $a$ is the set $\frac{\partial}{\partial_a}(E)$ of expressions computed as follows:*

$$\frac{\partial}{\partial_a}(\emptyset) = \frac{\partial}{\partial_a}(\varepsilon) = \frac{\partial}{\partial_a}(b) = \emptyset,\ \frac{\partial}{\partial_a}(a) = \{\varepsilon\},$$
$$\frac{\partial}{\partial_a}(F+G) = \frac{\partial}{\partial_a}(F) \cup \frac{\partial}{\partial_a}(G),\ \frac{\partial}{\partial_a}(F^*) = \frac{\partial}{\partial_a}(F) \cdot F^*,$$
$$\frac{\partial}{\partial_a}(F \cdot G) = \begin{cases} \frac{\partial}{\partial_a}(F) \cdot G \cup \frac{\partial}{\partial_a}(G) & \text{if } \varepsilon \in L(F), \\ \frac{\partial}{\partial_a}(F) \cdot G & \text{otherwise,} \end{cases}$$
$$\frac{\partial}{\partial_a}(\mathrm{f}(E_1, \ldots, E_k)) = \{\frac{d'}{d'_a}(\mathrm{f}(E_1, \ldots, E_k))\}$$

*Example 4.* This example illustrates the worst case of the natural extension, where partial derivative computation reduces to derivative computation. Let us consider the three expressions $F$, $G$ and $E$ defined by:

$$F = (a+b)^* a (a+b)^n,\ G = (\neg(\neg a \cap \neg b))^* a (a+b)^n,\ E = F \cap G.$$

According to Definition 1, $\frac{\partial}{\partial_a}(E) = \{\frac{d'}{d'_a}(E)\}$. Since the intersection operator is at the highest level of the syntax tree of $E$, for all word $w$ in $\Sigma^*$, the partial derivative of $E$ w.r.t. $w$ contains a unique derivated term: the derivative of $E$ w.r.t. $w$. The expression $F$ has an exponential number of dissimilar derivatives (*cf.* Example 2) and $\frac{d'}{d'_a}(F) \neq \emptyset \Leftrightarrow \frac{d'}{d'_a}(G) \neq \emptyset$. Consequently, $E$ has an exponential number of derivated terms.

### 3.2 Set of Sets Extension

The main strength of the partial derivative computation for a simple regular expression is to break the partial derivative of a sum into a union of derivated terms. A partial derivative is a set of simple regular expressions, the language of which is the union of the languages denoted by these expressions. As far as recognizers are concerned, the advantage is that the derivated term automaton is a non-deterministic one, and then it may be exponentially smaller than the deterministic derivative automaton. For extended regular expressions, Example 4 shows that the computation of a partial derivative is ineffective for any derivated term for which the highest operator is different from the sum.

We now show how to break the partial derivative of an intersection expression into a union of derivated terms. Consider the Example 4, where:

$$F = (a+b)^* a (a+b)^n,\ G = (\neg(\neg a \cap \neg b))^* a (a+b)^n \text{ and } E = F \cap G.$$

Let us set: $\mathcal{H}_1 = \{F, \Sigma^n, \ldots, \Sigma^{n-k+1}\}$ and $\mathcal{H}_2 = \{G, \Sigma^n, \ldots, \Sigma^{n-k+1}\}$.

In the natural extension, the partial derivative of $E$ w.r.t. $a^k$ contains a unique derivated term $T = (F + \Sigma^n + \ldots + \Sigma^{n-k+1}) \cap (G + \Sigma^n + \ldots + \Sigma^{n-k+1})$.

By distributivity of $\cap$ over $+$, the expression $T$ is equivalent to the expression $T' = \sum_{(H_1,H_2)\in\mathcal{H}_1\times\mathcal{H}_2} H_1 \cap H_2$.

Let us set: $\mathcal{K} = \{F, \Sigma^n, \ldots, \Sigma^0\} \times \{G, \Sigma^n, \ldots, \Sigma^0\}$. It can be checked that every derivated term of $E$ is equivalent to a sum of expressions $H_i \cap H_j$, where $(H_i, H_j)$ is an element in $\mathcal{K}$. There exist $(n+2)^2$ distinct expressions $H_i \cap H_j$, and each one will be transformed into a derivated term in our second extension. As a result of breaking the partial derivative of an intersection just like Antimirov breaks the partial derivative of a sum, the number of derivated terms can be exponentially smaller than the number of dissimilar derivatives. Finally, for an extended regular expression, a partial derivative is a set of derivated terms (the language of which is a union of languages), and a derivated term is a set of expressions (the language of which is an intersection of languages). Thus a partial derivative is a particular set of expression sets. We now study the properties of sets of expression sets.

Let $\mathbb{E}$ be a set of expression sets, with: $\mathbb{E} = \bigcup_{\mathcal{E}\in\mathbb{E}} \mathcal{E}$ and, for all $\mathcal{E} \in \mathbb{E}$, $\mathcal{E} = \bigcup_{E\in\mathcal{E}} E$. The language of $\mathbb{E}$ is defined by: $L(\mathbb{E}) = \bigcup_{\mathcal{E}\in\mathbb{E}} L(\mathcal{E})$ and, for all $\mathcal{E} \in \mathbb{E}$, $L(\mathcal{E}) = \bigcap_{E\in\mathcal{E}} L(E)$.

The following properties are satisfied by sets of expression sets.

**Lemma 1.** *Let $\mathbb{E}$ and $\mathbb{E}'$ be two sets of expression sets, $\mathcal{E}$ and $\mathcal{E}'$ be two expression sets, and $F$ be an expression.*

***(1)*** $L(\mathcal{E}) \cap L(\mathcal{E}') = L(\mathcal{E} \cup \mathcal{E}')$,
***(2)*** $L(\mathbb{E}) \cap L(\mathbb{E}') = \bigcup_{\mathcal{E}\in\mathbb{E},\mathcal{E}'\in\mathbb{E}'} L(\mathcal{E} \cup \mathcal{E}')$,
***(3)*** $\neg L(\mathbb{E}) = L(\bigcap_{\mathcal{E}\in\mathbb{E}} \sum_{E\in\mathcal{E}} \neg E)$,
***(4)*** $L(\mathbb{E}) \cdot L(F) = L(\sum_{\mathcal{E}\in\mathbb{E}}((\bigcap_{E\in\mathcal{E}} E) \cdot F))$.

*Proof.* According to definitions and properties of language operators:

**(1)** $L(\mathcal{E}) \cap L(\mathcal{E}') = \bigcap_{E\in\mathcal{E}} L(E) \cap \bigcap_{E'\in\mathcal{E}'} L(E')$
$= \bigcap_{E\in\mathcal{E}\cup\mathcal{E}'} L(E) \qquad = L(\mathcal{E} \cup \mathcal{E}')$

**(2)** $L(\mathbb{E}) \cap L(\mathbb{E}') = \bigcup_{\mathcal{E}\in\mathbb{E}} L(\mathcal{E}) \cap \bigcup_{\mathcal{E}'\in\mathbb{E}'} L(\mathcal{E}')$
$= \bigcup_{\mathcal{E}\in\mathbb{E},\mathcal{E}'\in\mathbb{E}'} L(\mathcal{E}) \cap L(\mathcal{E}') \quad = \bigcup_{\mathcal{E}\in\mathbb{E},\mathcal{E}'\in\mathbb{E}'} L(\mathcal{E} \cup \mathcal{E}')$

**(3)** $\neg L(\mathbb{E}) = \neg \bigcup_{\mathcal{E}\in\mathbb{E}} L(\mathcal{E}) \qquad = \neg \bigcup_{\mathcal{E}\in\mathbb{E}} \bigcap_{E\in\mathcal{E}} L(E)$
$= \bigcap_{\mathcal{E}\in\mathbb{E}} \bigcup_{E\in\mathcal{E}} \neg L(E) \; = \bigcap_{\mathcal{E}\in\mathbb{E}} \bigcup_{E\in\mathcal{E}} L(\neg E)$
$= \bigcap_{\mathcal{E}\in\mathbb{E}} L(\sum_{E\in\mathcal{E}} \neg E) = L(\bigcap_{\mathcal{E}\in\mathbb{E}} \sum_{E\in\mathcal{E}} \neg E)$

**(4)** $L(\mathbb{E}) \cdot L(F) = (\bigcup_{\mathcal{E}\in\mathbb{E}} L(\mathcal{E})) \cdot L(F) \qquad = (\bigcup_{\mathcal{E}\in\mathbb{E}} L(\mathcal{E}) \cdot L(F))$
$= (\bigcup_{\mathcal{E}\in\mathbb{E}} L(\bigcap_{E\in\mathcal{E}} E) \cdot L(F))$
$= (\bigcup_{\mathcal{E}\in\mathbb{E}} L((\bigcap_{E\in\mathcal{E}} E) \cdot F))$
$= L(\sum_{\mathcal{E}\in\mathbb{E}}((\bigcap_{E\in\mathcal{E}} E) \cdot F))$

□

Let $\mathbb{E}$ and $\mathbb{F}$ be two sets of expression sets and $G$ be an expression. The three following operators are defined: $\mathbb{E} \odot G = \bigcup_{\mathcal{E}\in\mathbb{E}}((\bigcap_{E\in\mathcal{E}} E) \cdot G)$, $\mathbb{E} \owedge \mathbb{F} = \bigcup_{\mathcal{E}\in\mathbb{E},\mathcal{F}\in\mathbb{F}}(\mathcal{E} \cup \mathcal{F})$, $\ominus\mathbb{E} = \owedge_{\mathcal{E}\in\mathbb{E}}(\bigcup_{E\in\mathcal{E}} \neg E)$. According to Lemma 1, it holds: $L(\mathbb{E} \odot F) = L(\mathbb{E}) \cdot L(F)$, $L(\mathbb{E} \owedge \mathbb{F}) = L(\mathbb{E}) \cap L(\mathbb{F})$ and $L(\ominus\mathbb{E}) = \neg L(\mathbb{E})$. These operators are used to define partial derivatives of an extended regular expression.

**Definition 2.** *The partial derivative of an extended regular expression $E$ w.r.t. a symbol $a$ is the set $\frac{\partial}{\partial_a}(E)$ of expression sets defined by:*

$$\textbf{(1)}\ \tfrac{\partial}{\partial_a}(\emptyset) = \tfrac{\partial}{\partial_a}(\varepsilon) = \tfrac{\partial}{\partial_a}(b) = \emptyset, \quad \textbf{(2)}\ \tfrac{\partial}{\partial_a}(a) = \{\{\varepsilon\}\}$$

$$\textbf{(3)}\ \tfrac{\partial}{\partial_a}(E+F) = \tfrac{\partial}{\partial_a}(E) \cup \tfrac{\partial}{\partial_a}(F), \quad \textbf{(4)}\ \tfrac{\partial}{\partial_a}(E^*) = \tfrac{\partial}{\partial_a}(E) \odot E^*$$

$$\textbf{(5)}\ \tfrac{\partial}{\partial_a}(E \cdot F) = \begin{cases} \tfrac{\partial}{\partial_a}(E) \odot F & \text{if } \varepsilon \notin L(E) \\ \tfrac{\partial}{\partial_a}(E) \odot F \cup \tfrac{\partial}{\partial_a}(F) & \text{otherwise.} \end{cases}$$

$$\textbf{(6)}\ \tfrac{\partial}{\partial_a}(\neg E) = \ominus(\tfrac{\partial}{\partial_a}(E)), \quad \textbf{(7)}\ \tfrac{\partial}{\partial_a}(E \cap F) = \tfrac{\partial}{\partial_a}(E) \owedge \tfrac{\partial}{\partial_a}(F)$$

**Proposition 1.** *Let $E$ be an extended regular expression over the alphabet $\Sigma$ and $a$ be a symbol in $\Sigma$. Then it holds: $L(\frac{\partial}{\partial_a}(E)) = a^{-1}(L(E))$.*

*Proof.* By induction on the structure of extended regular expressions.

According to [1], formulas **(1)** to **(3)** are satisfied.

$$\begin{aligned}
\textbf{(4)}\ L(\tfrac{\partial}{\partial_a}(E^*)) &= L(\tfrac{\partial}{\partial_a}(E) \odot E^*) \quad = L(\tfrac{\partial}{\partial_a}(E)) \cdot L(E^*) \\
&= a^{-1}(L(E)) \cdot L(E^*) = a^{-1}(L(E^*)) \\
\textbf{(5)}\ L(\tfrac{\partial}{\partial_a}(E \cdot F)) &= L(\tfrac{\partial}{\partial_a}(E) \odot F) \quad = L(\tfrac{\partial}{\partial_a}(E)) \cdot L(F) \\
&= a^{-1}(L(E)) \cdot L(F) = a^{-1}(L(E \cdot F)) \\
\textbf{(5')}\ L(\tfrac{\partial}{\partial_a}(E \cdot F)) &= L(\tfrac{\partial}{\partial_a}(E) \odot F \cup \tfrac{\partial}{\partial_a}(F)) \\
&= L(\tfrac{\partial}{\partial_a}(E)) \cdot L(F) \cup L(\tfrac{\partial}{\partial_a}(F)) \\
&= a^{-1}(L(E)) \cdot L(F) \cup a^{-1}(L(F)) = a^{-1}(L(E \cdot F)) \\
\textbf{(6)}\ L(\tfrac{\partial}{\partial_a}(\neg E)) &= L(\ominus \tfrac{\partial}{\partial_a}(E)) \quad = \neg L(\tfrac{\partial}{\partial_a}(E)) \\
&= \neg(a^{-1}(L(E))) = a^{-1}(\neg(L(E))) \\
\textbf{(7)}\ L(\tfrac{\partial}{\partial_a}(E \cap F)) &= L(\tfrac{\partial}{\partial_a}(E) \owedge \tfrac{\partial}{\partial_a}(F)) \quad = L(\tfrac{\partial}{\partial_a}(E)) \cap L(\tfrac{\partial}{\partial_a}(F)) \\
&= a^{-1}(L(E)) \cap a^{-1}(L(F)) = a^{-1}(L(E \cap F))
\end{aligned}$$

□

Every partial derivative of an extended regular expression $E$ w.r.t. a symbol is a set $\mathbb{E}$ in which each element $\mathcal{E}$ is a derivated term of $E$. The partial derivative of a derivated term is computed as follows.

**Definition 3.** *Let $E$ be an extended regular expression over an alphabet $\Sigma$ and $a$ be a symbol in $\Sigma$. Let $\mathcal{E}$ be a derivated term of $E$. Then:*

$$\tfrac{\partial}{\partial_a}(\mathcal{E}) = \owedge_{E_k \in \mathcal{E}}(\tfrac{\partial}{\partial_a}(E_k)).$$

**Proposition 2.** *Let $E$ be an extended regular expression over an alphabet $\Sigma$ and $a$ be a symbol in $\Sigma$. Let $\mathcal{E}$ be a derivated term of $E$. Then:*

$$L(\tfrac{\partial}{\partial_a}(\mathcal{E})) = a^{-1}L(\mathcal{E}).$$

*Proof.* By equality of sets:

$$\begin{aligned}
L(\tfrac{\partial}{\partial_a}(\mathcal{E})) &= L(\owedge_{E_k \in \mathcal{E}} \tfrac{\partial}{\partial_a}(E_k)) \quad = \bigcap_{E_k \in \mathcal{E}} L(\tfrac{\partial}{\partial_a}(E_k)) \\
&= \bigcap_{E_k \in \mathcal{E}} a^{-1}(L(E_k)) = a^{-1}(L(\bigcap_{E_k \in \mathcal{E}} E_k)) = a^{-1}(L(\mathcal{E}))
\end{aligned}$$

□

The partial derivative of an extended regular expression $E$ w.r.t. a word $aw$ in $\Sigma^+$ is defined by $\frac{\partial}{\partial_{aw}}(E) = \bigcup_{\mathcal{E}' \in \frac{\partial}{\partial_a}(E)} \frac{\partial}{\partial_w}(\mathcal{E}')$.

**Proposition 3.** *Let $E$ be an extended regular expression over an alphabet $\Sigma$ and $aw$ be a word in $\Sigma^+$. Then it holds: $(aw)^{-1}(L(E)) = L(\frac{\partial}{\partial_{aw}}(E))$.*

*Proof.* By induction on the length of $w$:

$$L(\tfrac{\partial}{\partial_u}(E)) = L(\tfrac{\partial}{\partial_{aw'}}(E)) = L(\tfrac{\partial}{\partial_{w'}}(\tfrac{\partial}{\partial_a}(E))).$$

By the recurrence hypothesis: $L(\frac{\partial}{\partial_{w'}}(\frac{\partial}{\partial_a}(E))) = w'^{-1}(L(\frac{\partial}{\partial_a}(E)))$.

According to Proposition 1, $L(\frac{\partial}{\partial_a}(E)) = a^{-1}(L(E))$.

Consequently, $L(\frac{\partial}{\partial_u}(E)) = w'^{-1}(a^{-1}(L(E))) = u^{-1}(L(E))$. □

## 4 The Derivated Term Automaton

Let $E$ be an extended regular expression, $\mathcal{E}$ be a derivated term of $E$, and $\mathbb{E}$ be a partial derivative of $E$ such that $\mathcal{E} \in \mathbb{E}$. Every expression $E_k$ in $\mathcal{E}$ is called a *derivated expression* of $E$. This relation is noted $E_k \circledcirc \mathbb{E}$. According to Definition 3, for all symbol $a \in \Sigma$ and for all derivated expression $E_k \circledcirc \frac{\partial}{\partial_a}(\mathcal{E})$, there exists a derivated expression $F_k$ in $\mathcal{E}$ such that $E_k \circledcirc \frac{\partial}{\partial_a}(F_k)$. Consequently, in order to compute the set of derivated terms of $E$, it is sufficient to compute the partial derivatives of derivated expressions and then to combine them with the operator $\circledcap$.

The set $\mathcal{D}'_E = \{\mathcal{E} \mid \exists w \in \Sigma^*, \mathcal{E} \in \frac{\partial}{\partial_w}(E)\}$ is called the *set of derivated terms* of $E$. The *derivated term automaton of* $E$, $A = (\Sigma, Q, q_0, F, \delta)$, is defined similarly as for a simple expression: $Q = \mathcal{D}'_E$, $q_0 = \{E\}$, $F = \{\mathcal{E}' \in \mathcal{D}'_E \mid \varepsilon \in L(\mathcal{E}')\}$, and $\delta = \{(\mathcal{F}, a, \mathcal{G}) \in Q \times \Sigma \times Q \mid \mathcal{G} \in \frac{\partial}{\partial_a}(\mathcal{F})\}$.

*Example 5.* Let us consider the expression $E = F \cap G$ in Example 4. The derivated terms of $E$ are computed as follows:

$$\begin{aligned}
\tfrac{\partial}{\partial_a}(E) &= \tfrac{\partial}{\partial_a}(F) \circledcap \tfrac{\partial}{\partial_a}(G) & \tfrac{\partial}{\partial_b}(E) &= \{\{F, G\}\} \\
&= \{\{F\}, \{\Sigma^n\}\} \circledcap \{\{G\}, \{\Sigma^n\}\} & & \\
&= \{\{F, G\}, \{F, \Sigma^n\}, \{G, \Sigma^n\}, \{\Sigma^n\}\} & & \\
\tfrac{\partial}{\partial_a}(F) &= \{\{F\}, \{\Sigma^n\}\} & \tfrac{\partial}{\partial_b}(F) &= \{\{F\}\} & \tfrac{\partial}{\partial_a}(\{\Sigma^k\}) &= \{\{\Sigma^{k-1}\}\} \\
\tfrac{\partial}{\partial_a}(G) &= \{\{G\}, \{\Sigma^n\}\} & \tfrac{\partial}{\partial_b}(G) &= \{\{G\}\} & \tfrac{\partial}{\partial_b}(\{\Sigma^k\}) &= \{\{\Sigma^{k-1}\}\}
\end{aligned}$$

The set of derivated terms is: $\{F \cap G\} \cup \{F, \Sigma^n, \ldots, \varepsilon\} \times \{G, \Sigma^n, \ldots, \varepsilon\}$.

There are $k$ derivated terms, where:

$$k = 1 + (n+2) + (n+2) + (n+1) + \ldots + 2 = \tfrac{(n+2)(n+4)}{2}.$$

The derivated term automaton of $E$ is represented in Figure 3.

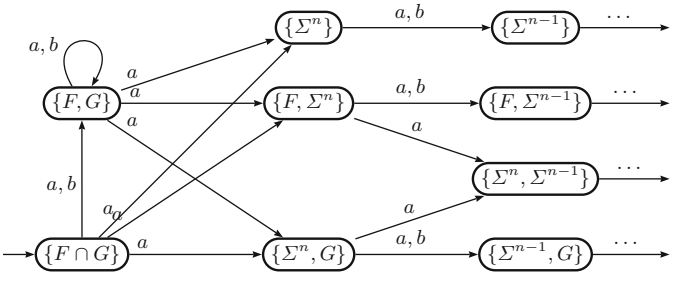


**Fig. 3.** The derivated term automaton of $E = F \cap G$

**Proposition 4.** *Let $E$ be an extended regular expression and $A$ be its derivated term automaton. Then it holds: $L(A) = L(E)$.*

*Proof.* By construction, $w \in L(A) \Leftrightarrow w \in L(\frac{\partial}{\partial_w}(E)) \Leftrightarrow w \in L(E)$. □

The derivated term automaton of an extended regular expression $E$ may have an exponential number of states with respect to the width of $E$. It is the case for the expression $F = \neg E_n$ where $E_n = (a+b)^* a (a+b)^n$. It can be checked however that in this case, the derivated term automaton is not minimal. This situation is consistent with the following proposition that addresses the complexity of the size of the automata that recognize the language $L(\neg E)$, where $E$ is a simple regular expression.

**Proposition 5.** ***(1)*** *For every simple regular expression $E$ over an alphabet $\Sigma$, it is possible to compute an automaton $D$ such that:*

$$L(D) = L(\neg E) \text{ and } O(|D|) = O(2^{|E|+1}).$$

***(2)*** *There exists a family of expressions $(r_n)_{n\in\mathbb{N}}$ such that every automaton recognizing $L(\neg r_n)$ has an exponential size with respect to $|r_n| + 1$.*

*Proof.* Proof of this proposition is based on the following known results:

**(a)** For every simple regular expression $E$, a non-deterministic automaton $G$ can be computed such that $L(G) = L(E)$ and $|G| = |E| + 1$ [9,12] or $|G| \leq |E| + 1$ [1]. **(b)** For every non-deterministic automaton $A$, there exists a deterministic and complete automaton $D$ such that $L(D) = L(\neg E)$ and $|D| \leq 2^{|A|}$ [15]. **(c)** For every automaton $A$ over an alphabet $\Sigma$, a simple regular expression $E$ such that $L(A) = L(E)$ and $O(|E|) = O(|A| \times |\Sigma| \times 4^{|A|})$ [12,6] can be computed. **(d)** For $\Sigma$ an alphabet of size 4, for all $n \in \mathbb{N}$, there exists a simple regular expression $r_n$ which size is $O(n)$ such that for all simple regular expression $r$ such that $L(r) = \Sigma^* \setminus L(r_n)$, $|r| \geq 2^{2^n}$ [8,7].

**(1)** From **(a)** and **(b)**.

**(2)** According to **(1)**, for every simple regular expression $E$, one can construct an automaton $D$ such that $L(D) = L(\neg E)$ and $|D|$ is at most exponential w.r.t. $|E| + 1$. According to **(c)**, $D$ can be converted into an expression $F$ such that $L(D) = L(\neg E) = L(F)$ and $|F|$ is at most exponential w.r.t. $2^{|E|+1}$. Following [8,7], let us set $E = r_n$. Then $|D|$ must be exponential w.r.t. $|E| + 1$ and $|F|$ must be exponential w.r.t. $2^{|E|+1}$ so that condition **(d)** be satisfied. Therefore any automaton recognizing $L(\neg E_n)$ has at least an exponential size w.r.t. $|E_n| + 1$. □

## 5 The Partial Derivative Automaton

As in the case of a simple regular expression, the language of an extended regular expression $E$ is recognized by a deterministic automaton the states of which are the partial derivatives of $E$; moreover this automaton can be computed by applying the subset construction to the derivated term automaton of $E$.

**Definition 4.** *Let $E$ be an extended regular expression over the alphabet $\Sigma$.* The partial derivative automaton *of $E$ is the deterministic automaton $A' = (\Sigma, Q', q'_0, F', \delta')$ defined by $Q' = \{\frac{\partial}{\partial_w}(E) \mid w \in \Sigma^*\}$, $q'_0 = \{\{E\}\}$, $F' = \{G' \in Q' \mid \varepsilon \in L(G')\}$, $\delta' = \{(G', a, H') \mid \frac{\partial}{\partial_a}(G') = H'\}$.*

**Proposition 6.** *Let $E$ be an extended regular expression. The partial derivative automaton $A'$ of $E$ is obtained by applying the subset construction to the derivated term automaton $A$ of $E$.*

*Proof.* The proof is the same as in the simple regular expression case. Let $A = (\Sigma, Q, q_0, F, \delta)$ and $A' = (\Sigma, Q', q'_0, F', \delta')$. By definition of $A'$, for all $w \in \Sigma^*$, the state $q'_0 \cdot w$ is the partial derivative of $E$ w.r.t. $w$. Let us consider the automaton $D = (\Sigma, S, s_0, T, \cdot)$ obtained by applying the subset construction to the automaton $A$. For all word $w \in \Sigma^*$, the deterministic state $s_0 \cdot w$ is associated with the subset $\delta(q_0, w)$ of states of $Q$. By definition of $A$, this subset is equal to the union of the derivated terms of $E$ w.r.t. $w$, that is to the partial derivative of $E$ w.r.t. $w$. Hence, applying the subset construction to the automaton $A$ yields the automaton $A'$.

**Proposition 7.** *Let $E$ be an extended regular expression, $B$ be the derivative automaton of $E$ and $A'$ be the partial derivative automaton of $E$. The automata $B$ and $A'$ generally are not comparable.*

*Proof.* A state of $B$ is a derivative of $E$ while a state of $A'$ is a partial derivative of $E$. Thus it seems natural to define a morphism (for example from $B$ to $A'$) by associating the state $\frac{d'}{d'_u}(E)$ to the state $\frac{\partial}{\partial_u}(E)$. However this correspondence only results in a mapping from $B$ to $A'$ if for all $v \in \Sigma^*$ such that $\frac{d'}{d'_u}(E) = \frac{d'}{d'_v}(E)$, we also have $\frac{\partial}{\partial_u}(E) = \frac{\partial}{\partial_v}(E)$. The reasoning is similar for a morphism from $A'$ to $B$. We now exhibit extended regular expressions such that there exists no morphism either from $B$ to $A'$ or from $A'$ to $B$.

**(1)** Let us consider the expression $E = a(a+\varepsilon)(ba+b)^* + (ba+b)^*$ over the alphabet $\Sigma = \{a, b\}$. It holds:

$$\frac{d'}{d'_a}(E) = (a+\varepsilon)(ba+b)^* \qquad \frac{d'}{d'_b}(E) = (a+\varepsilon)(ba+b)^*$$
$$\frac{\partial}{\partial_a}(E) = \{\{(a+\varepsilon)(ba+b)^*\}\} \quad \frac{\partial}{\partial_b}(E) = \{\{a(ba+b)^*\}, \{(ba+b)^*\}\}.$$

The derivatives of $E$ w.r.t. $a$ and $b$ are equal, although the partial derivatives of $E$ w.r.t. $a$ and $b$ are not equal.

**(2)** Let us consider the expression $F = (ba^* \cap ba^*)b + aa^*b$ over the alphabet $\Sigma = \{a, b\}$. It holds:

$$\frac{d'}{d'_a}(E) = a^*b \qquad \frac{d'}{d'_b}(E) = (a^* \cap a^*)b$$
$$\frac{\partial}{\partial_a}(E) = \{\{a^*b\}\} \quad \frac{\partial}{\partial_b}(E) = \{\{a^*b\}\}.$$

The partial derivatives of $E$ w.r.t. $a$ and $b$ are equal, although the derivatives of $E$ w.r.t. $a$ and $b$ are not equal.

## 6 Conclusion

Thanks to an appropriate definition of partial derivatives of complementation and intersection operations, we have generalized the partial derivative computation to extended regular expressions. As a result, and generalizing Antimirov algorithm, we have designed an algorithm for converting an extended regular expression into a non-deterministic automaton. To our knowledge, it is the first algorithm computing such a non-deterministic automaton. Notice that the other boolean operations can be expressed through complementation and intersection operations, for example the set difference $L(E \setminus F) = L(E) \cap \neg L(F)$ or the symmetrical difference $L(E \Delta F) = (L(E) \cap \neg L(F)) \cup (L(F) \cap \neg L(E))$. Thus the partial derivatives of the other boolean operations can easily be processed using the formulae of complementation and intersection operations. For a regular expression with $n$ positions, the number of states of the derivated term automaton is bounded by $n+1$ if the expression is a simple one, whereas it has a worst case exponential complexity (and this bound is tight) if the expression is an extended one. It is an open question whether there exists an extension of the notion of position in an extended regular expression that would generalize the relation between positions and derivated terms of a simple regular expression.

## References

1. Antimirov, V.: Partial derivatives of regular expressions and finite automaton constructions. Theoretical Computer Science 155, 291–319 (1996)
2. Berry, G., Sethi, R.: From regular expressions to deterministic automata. Theoretical Computer Science 48(1), 117–126 (1986)
3. Brzozowski, J.A.: Derivatives of regular expressions. Journal of the Association for Computing Machinery 11(4), 481–494 (1964)
4. Champarnaud, J.M., Ponty, J.L., Ziadi, D.: From regular expressions to finite automata. International Journal of Computational Mathematics 72, 415–431 (1999)
5. Champarnaud, J.M., Ziadi, D.: Canonical derivatives, partial derivatives, and finite automaton constructions. Theoretical Computer Science 239(1), 137–163 (2002)
6. Ellul, K., Krawetz, B., Shallit, J., Wang, M.: Regular expressions: New results and open problems. Journal of Automata, Languages and Combinatorics 10(4), 407–437 (2005)
7. Gelade, W.: Succinctness of regular expressions with interleaving, intersection and counting. Theoretical Computer Science 411(31-33), 2987–2998 (2010)
8. Gelade, W., Neven, F.: Succinctness of the complement and intersection of regular expressions. In: Albers, S., Weil, P. (eds.) STACS. Dagstuhl Seminar Proceedings, vol. 08001, pp. 325–336 (2008)
9. Glushkov, V.M.: The abstract theory of automata. Russian Mathematical Surveys 16, 1–53 (1961)
10. Kleene, S.: Representation of events in nerve nets and finite automata. Automata Studies Annual Mathematical Studies 34, 3–41 (1956)
11. Leiss, E.: Constructing a finite automaton for a given regular expression. SIGACT News 12, 81–87 (1980)

12. McNaughton, R.F., Yamada, H.: Regular expressions and state graphs for automata. IEEE Transactions on Electronic Computers 9, 39–57 (1960)
13. Myhill, J.: Finite automata and the representation of events. Wright Air Development Command Technical Report 57-624, 112–137 (1957)
14. Nerode, A.: Linear automata transformation. In: Proceedings of AMS, vol. 9, pp. 541–544 (1958)
15. Rabin, M.O., Scott, D.: Finite automata and their decision problems. IBM Journal of Research and Development 3(2), 115–125 (1959)