ELSEVIER

Theoretical Computer Science 289 (2002) 137–163

Theoretical Computer Science

www.elsevier.com/locate/tcs

Canonical derivatives, partial derivatives and finite automaton constructions

J.-M. Champarnaud, D. Ziadi

Université de Rouen, LIFAR, Place E Blondel, F-76821 Mont-Saint-Aignan Cedex, France

Received July 1999; received in revised form April 2001; accepted May 2001
Communicated by M. Nivat
En hommage à Valentin Antimirov

Abstract

Let E be a regular expression. Our aim is to establish a theoretical relation between two well-known automata recognizing the language of E, namely the position automaton $\mathscr{P}_E$ constructed by Glushkov or McNaughton and Yamada, and the equation automaton $\mathscr{E}_E$ constructed by Mirkin or Antimirov. We define the notion of c-derivative (for canonical derivative) of a regular expression E and show that if E is linear then two Brzozowski's derivatives of E are aci-similar if and only if the corresponding c-derivatives are identical. It allows us to represent the Berry–Sethi's set of continuations of a position by a unique c-derivative, called the c-continuation of the position. Hence the definition of $\mathscr{C}_E$, the c-continuation automaton of E, whose states are pairs made of a position of E and of the associated c-continuation. If states are viewed as positions, $\mathscr{C}_E$ is isomorphic to $\mathscr{P}_E$. On the other hand, a partial derivative, as defined by Antimirov, is a class of c-derivatives for some equivalence relation, thus $\mathscr{C}_E$ reduces to $\mathscr{E}_E$. Finally $\mathscr{C}_E$ makes it possible to go from $\mathscr{P}_E$ to $\mathscr{E}_E$, while this cannot be achieved directly (from the state graphs). These theoretical results lead to an $O(|E|^2)$ space and time algorithm to compute the equation automaton, where $|E|$ is the size of the expression. This is the complexity of the most efficient constructions yielding the position automaton, while the size of the equation automaton is not greater and generally much smaller than the size of the position automaton.

Keywords: Regular expressions; Finite automata; Derivatives

1. Introduction

This work first enlightens the relation between two fundamental automata, both nondeterministic and obtained from a regular expression:

E-mail addresses: champarnaud@dir.univ-rouen.fr (J.-M. Champarnaud), ziadi@dir.univ-rouen.fr (D. Ziadi).

PII: S0304-3975(01)00267-5

(1) the position automaton, which is constructed by the classical algorithms of Glushkov [8] and of McNaughton and Yamada [11],
(2) the equation automaton, which is the result of the method of Mirkin [12] based on the notion of prebase, and of the algorithm of Antimirov [2] based on the notion of partial derivative.

Our theoretical contribution is the definition and the construction of a third automaton, which is both isomorphic to the position automaton by a natural projection and isomorphic to the equation automaton by a quotient construction. Let us point out that there is no state equivalence (in terms of graphs) leading to the latter isomorphism.

We extend the study of Berry and Sethi [4] who have characterized the Brzozowski's derivatives [5] of a linear expression. We introduce the notion of canonical derivative, c-derivative for short, of a regular expression E, in order to compute canonical representatives of the sets of the aci-similar Brzozowski's derivatives of a linear expression. Thus the Berry–Sethi's set of continuations of a position is represented by a unique c-derivative, called the c-continuation of the position.

We give a constructive definition of c-derivatives and prove several facts, in particular: (1) the Brzozowski's derivatives of a linear expression w.r.t. two words u and v are aci-similar if and only if the c-derivatives w.r.t. u and v are identical; (2) any non-zero c-derivative of E is either 1 or a subexpression of E or a concatenation of several subexpressions; (3) there exists a surjective mapping from the set of the non-zero c-derivatives of the linearized version of a regular expression E onto the set of the partial derivatives of E.

These theoretical results lead to the definition of the c-continuation automaton, whose states are pairs made of a position added with the associated c-continuation, and they allow us to prove that this automaton can be viewed as the position automaton and that it reduces to the equation automaton.

This work secondly yields the following algorithmic result: the equation automaton of a regular expression E can be computed with an $O(|E|^2)$ space and time complexity. Let us recall the $O(\|E\|^2|E|^2)$ worst-case time complexity [1] of the algorithm described by Antimirov in [2]. It is well known that the number of states of the equation automaton is less than or equal to the number of states of the position automaton (and can be much smaller). Hence the interest of the new algorithm we present here: (1) it has the same complexity as the most efficient constructions of the position automaton, and (2) its result has at worst as many states as the position automaton.

Section 2 recalls some useful notations, definitions and classical results of automaton theory.

In Section 3 we give the definition of the c-derivative of a regular expression and we present the properties of the c-derivatives of linear expressions, in relation with the

[1] Actually, this complexity only covers the computation of the set of states. As explained in Section 7, the whole Antimirov construction is in $O(\|E\|^3|E|^2)$ time complexity.

Brzozowski's derivatives and the Berry–Sethi's continuations. As a result we provide the definition of the c-continuation automaton of a regular expression.

In Section 4 we show that the c-continuation automaton is isomorphic to the position automaton, as far as its states, which are pairs made of a position added with the corresponding c-continuation, are viewed as positions.

Section 5 recalls the definition and properties of partial derivatives and the definition of the equation automaton as constructed by the partial derivative algorithm. The relation between c-derivatives and partial derivatives is established.

In Section 6 we show that the c-continuation automaton reduces to the equation automaton, two states being equivalent if and only if the associated c-continuations (defined over the position alphabet) are identical on the initial alphabet.

Section 7 presents the sketch of the Algorithm *CtoE*3 which builds the c-continuation automaton with an $O(\|E\| \cdot |E|^2)$ space and time complexity. It is based on specific procedures which compute (1) the list of the c-continuations associated to positions, (2) the equivalence classes on the set of c-continuations, and (3) the set of transitions of the c-continuation automaton. Some refinements leading to an $O(|E|^2)$ space and time complexity construction of the equation automaton are also provided. These algorithmic results are compared to previous ones and discussed.

2. Preliminaries

We first recall some definitions, basic notions and terminology concerning regular languages, regular expressions, finite automata and derivatives. For further details about these topics, we refer to classical books [3, 9] or handbooks [14].

2.1. Regular expressions and languages

Let Σ be a non-empty finite set of symbols, called the *alphabet*. The set of all the words over Σ is denoted by Σ^*. The empty word is denoted by ε. A *language* over Σ is a subset of Σ^*.

Regular expressions over an alphabet Σ and *regular languages* they denote are inductively defined as follows:

(1) 0 is a regular expression denoting the language $L(0) = \emptyset$.

(2) 1 is a regular expression denoting the language $L(1) = \{\varepsilon\}$.

(3) a, for all $a \in \Sigma$, is a regular expression denoting the language $L(a) = \{a\}$.

Let F (resp. G) be a regular expression denoting the language $L(F)$ (resp. $L(G)$). Then we have:

(4) $(F + G)$ is a regular expression denoting the language $L(F + G) = L(F) \cup L(G)$.

(5) $(F \cdot G)$ is a regular expression denoting the language $L(F \cdot G) = L(F)L(G)$.

(6) (F^*) is a regular expression denoting the language $L(F^*) = (L(F))^*$.

A regular expression over the alphabet Σ is a term of the algebra defined over the set $\Sigma \cup \{0, 1\}$, with the symbols of function $*, +, \cdot$, where $*$ is unary and $+$ and $\cdot$ are

binary. Properties of the constants 0 and 1, and of the operators $*$, $+$ and $\cdot$ lead to identities on this algebra. The following equations are classically used: $0+E=E=E+0$, $1\cdot E=E=E\cdot 1$, $0\cdot E=0=E\cdot 0$. Associativity, commutativity and idempotency properties of the $+$ operation, called aci-rules, are captured by the notion of similarity of two expressions introduced by Brzozowski [5]: two regular expressions F and G are said to be aci-similar ($F\sim_{\text{aci}} G$) if and only if they reduce to the same expression by applying aci-rules. We shall write: $F\equiv G$ if two regular expressions are identical (following Mirkin [12], E and F "graphically coincide").

We shall use the following definition:

$$\lambda(E)=\begin{cases}1 & \text{if } \varepsilon\in L(E),\\ 0 & \text{otherwise.}\end{cases}$$

2.2. Finite automata and recognizable languages

Let Σ be a finite alphabet. A *finite automaton* over Σ is a 5-tuple $\mathcal{M}=(Q,\Sigma,I,T,E)$ where Q is a set of *states*, I is a subset of Q whose elements are the *initial states*, T is a subset of Q whose elements are the *final states*, and E is a subset of the Cartesian product $Q\times\Sigma\times Q$ whose elements are the *edges*.

Let $\mathcal{M}=(Q,\Sigma,I,T,E)$ be an automaton. The automaton $\mathcal{M}$ is *deterministic* if there is only one initial state and if for all $(q,a)\in Q\times\Sigma$ there is at most one state q' such that $(q,a,q')\in E$. $\mathcal{M}$ is said to be a *DFA* if it is deterministic and a *NFA* otherwise. We shall write $\mathcal{M}=(Q,\Sigma,i,T,E)$ for an automaton with a unique initial state i. Edges are also called *transitions* and can be represented by a transition function δ from $Q\times\Sigma$ to Q when $\mathcal{M}$ is a DFA, and from $Q\times\Sigma$ to 2^Q when $\mathcal{M}$ is a NFA.

A *path* of $\mathcal{M}$ is a sequence (q_i,a_i,q_{i+1}), $i=1,\ldots,n$, of consecutive edges. Its *label* is the word $w=a_1a_2\ldots a_n$. A word $w=a_1a_2\ldots a_n$ is *recognized* by the automaton $\mathcal{M}$ if there exists a path with label w such that $q_1\in I$ and $q_{n+1}\in T$. The language *recognized* by the automaton $\mathcal{M}$ is the set of words which it recognizes. A language L is *recognizable* if and only if there exists a finite automaton whose language is L. Kleene's theorem [10] states that the set $Rat(\Sigma^*)$ of regular languages over Σ and the set $Reg(\Sigma^*)$ of recognizable languages over Σ are equal.

2.3. Linear expressions

Let E be a regular expression over the alphabet Σ_E. The *alphabetic width* of E, denoted by $\|E\|$, is the number of occurrences of symbols in E. E is said to be *linear* over Σ_E if and only if every symbol of Σ_E occurs (at most) one time in E. For all j in $[1,\|E\|]$, if x is the jth occurrence of symbol in E, then the pair (x,j) is called a *position* of E. By abuse of notation (x,j) is written x_j. The set of positions of E is denoted by Pos_E: $Pos_E=\{x_j \mid x\in\Sigma_E, j\in[1,\|E\|]\}$.

Let Σ' be an alphabet of size $\|E\|$, and $\mu_{\Sigma'}$ a one-to-one mapping from Pos_E onto Σ'. The expression E' over Σ' obtained from E by replacing symbol x of rank j by $\mu_{\Sigma'}(x_j)$, for all j in $[1,\|E\|]$, is called *a linearization* of E. Any linearization E' is linear over

Σ'. The *linearized version* of E, denoted by $\bar{E}$, is its linearization over Pos_E. Let us take for example $E = a \cdot (a+b)+(a+b) \cdot (1+b)$. We have: $Pos_E = \{a_1, a_2, b_3, a_4, b_5, b_6\}$, and $\bar{E} = a_1 \cdot (a_2 + b_3) + (a_4 + b_5) \cdot (1 + b_6)$.

Let h_E be the alphabetic mapping from Pos_E to Σ_E such that $h_E(x_i) = x$, $\forall i \in [1, \|E\|]$, and $h_E(\bar{E}) \equiv E$. Then, for every linearization E' over Σ', $h_{E'} = h_E \circ \mu_{\Sigma'}^{-1}$ is a mapping from Σ' to Σ_E such that $h_{E'}(E') \equiv E$. In our current example, we have: $h_E(a_1) = h_E(a_2) = h_E(a_4) = a$ and $h_E(b_3) = h_E(b_5) = h_E(b_6) = b$.

For subexpressions F of E, we shall consider the two following linearizations. The first one is the linearized version of the expression F whose alphabet is $\Sigma_F \subseteq \Sigma_E$. In our current example, $F = (a+b) \cdot (1+b)$ is a subexpression of E. The linearized version of F is the expression $\bar{F} = (a_1 + b_2) \cdot (1 + b_3)$ over the alphabet $Pos_F = \{a_1, b_2, b_3\}$. The alphabetic mapping h_F is such that $h_F(a_1) = a$, $h_F(b_2) = h_F(b_3) = b$ and $h_F(\bar{F}) = (a + b) \cdot (1 + b) = F$.

The second one is the subexpression of $\bar{E}$ which corresponds to F. Let us denote by $Pos_E(F)$ the set of positions of E which occur in F. The subexpression $\overline{F_E}$ of $\bar{E}$ whose set of symbols is $Pos_E(F)$ is a linearization of F. More formally, $\overline{F_E}$ is derived from F as follows, where H is a subexpression of E:

$$F = 0 \Rightarrow \overline{F_E} = 0,$$

$$F = 1 \Rightarrow \overline{F_E} = 1,$$

$$F = a, Pos_E(F) = \{a_i\} \Rightarrow \overline{F_E} = a_i,$$

$$H = F + G \Rightarrow \overline{H_E} = \overline{F_E} + \overline{G_E},$$

$$H = F \cdot G \Rightarrow \overline{H_E} = \overline{F_E} \cdot \overline{G_E},$$

$$H = F^* \Rightarrow \overline{H_E} = \overline{F_E}^*.$$

The mapping h_E maps $Pos_E(F)$ onto Σ_F and is such that: $h_E(\overline{F_E}) = F$. In our current example, if $F = (a + b) \cdot (1 + b)$, then $\overline{F_E} = (a_4 + b_5) \cdot (1 + b_6)$ over the alphabet $Pos_E(F) = \{a_4, b_5, b_6\}$. The mapping h_E is such that $h_E(a_4) = a$, $h_E(b_5) = h_E(b_6) = b$ and $h_E(\overline{F_E}) = (a + b) \cdot (1 + b) = F$.

Let $Pos_F = \{x_j \mid x \in \Sigma_E,\ j \in [1, ||F||]\}$. Let y_{k+1} be the first position of E occurring in F. We have $Pos_E(F) = \{y_{k+j} \mid y \in \Sigma_E,\ j \in [1, ||F||]\}$ and $y = h_E(y_{k+j}) = h_F(x_j) = x$, $\forall j \in [1, ||F||]$. The one-to-one mapping μ_F from $Pos_E(F)$ onto Pos_F is defined by $\mu_F(y_{k+j}) = x_j$, $\forall j \in [1, ||F||]$. We have: $\mu_F(\overline{F_E}) = \bar{F}$. In our current example, $\mu_F(\overline{F_E}) = \mu_F((a_4 + b_5) \cdot (1 + b_6)) = (a_1 + b_2) \cdot (1 + b_3) = \bar{F}$.

2.4. Word derivatives

Word derivatives of regular expressions have been introduced by Brzozowski in [5].

Definition 1 (*symbol derivative*). Given a regular expression E and a symbol a, the derivative of E w.r.t. a, written $a^{-1}E$, is recursively defined on the structure of E as

follows: [2]

$$a^{-1}0 = 0, \tag{1}$$

$$a^{-1}1 = 0, \tag{2}$$

$$a^{-1}x = 1 \quad \text{if } a = x \quad 0 \text{ otherwise}, \tag{3}$$

$$a^{-1}(F + G) = a^{-1}F + a^{-1}G, \tag{4}$$

$$a^{-1}(F \cdot G) = \begin{cases} a^{-1}F \cdot G & \text{if } \lambda(F) = 0, \\ a^{-1}F \cdot G + a^{-1}G & \text{otherwise}, \end{cases} \tag{5}$$

$$a^{-1}(F^*) = a^{-1}F \cdot F^*. \tag{6}$$

Definition 2 (*word derivative*). Derivatives w.r.t. a symbol can be extended to derivatives w.r.t. a word $u = u_1 \dots u_n$ in the following way:

$$\varepsilon^{-1}E = E, \tag{7}$$

$$(u_1 \dots u_n)^{-1}E = (u_2 \dots u_n)^{-1}(u_1^{-1}E). \tag{8}$$

More generally, we shall use the equation

$$(ps)^{-1}E = s^{-1}(p^{-1}E), \tag{9}$$

which holds for every factorization $u = ps$.

Let $\mathscr{D}(E)$ be the quotient of the set of all derivatives of a regular expression E modulo the aci-equivalence relation.

Theorem 1 (Brzozowski [5]). *Let E be a regular expression. The set $\mathscr{D}(E)$ of derivatives modulo aci-equivalence is finite.*

This result leads to the definition of the Brzozowski deterministic automaton $\mathscr{B}_E$, whose states are the derivatives of E modulo aci-equivalence. This automaton recognizes $L(E)$ [5].

Definition 3 (*Brzozowski automaton*). The Brzozowski automaton of a regular expression E, $\mathscr{B}_E = (Q, \Sigma, i, T, \delta)$, is defined by

- $Q = \mathscr{D}(E)$,
- $i = [E]$,
- $T = \{[d] \in \mathscr{D}(E) \mid \lambda(d) = 1\}$,
- $\delta([d], a) = [a^{-1}(d)]$, $\forall [d] \in Q$ and $\forall a \in \Sigma$.

[2] Eq. (5) takes into account the remark of Antimirov [2] about original equations.

3. From word derivatives to c-derivatives

We first recall specific properties of word derivatives of linear expressions leading to the definition of Berry–Sethi's continuations. We then introduce the notion of c-derivative which makes it possible to compute a canonical representative from a set of continuations.

3.1. Word derivatives of a linear expression

As far as linear expressions are concerned, subexpressions F and G have disjoint alphabets, hence a specific computation of word derivatives in the case when E is a sum or a concatenation.

Proposition 1. *For a linear expression E, the derivative of E by a symbol a can be computed by the following equations*:

$$a^{-1}0 = 0, \tag{10}$$

$$a^{-1}1 = 0, \tag{11}$$

$$a^{-1}x = 1 \quad if\ a = x \quad 0\ otherwise, \tag{12}$$

$$a^{-1}(F+G) = \begin{cases} a^{-1}F & if\ a \in \Sigma_F, \\ a^{-1}G & if\ a \in \Sigma_G, \\ 0 & otherwise, \end{cases} \tag{13}$$

$$a^{-1}(F\cdot G) = \begin{cases} a^{-1}F\cdot G & if\ a \in \Sigma_F, \\ \lambda(F)\cdot a^{-1}G & if\ a \in \Sigma_G, \\ 0 & otherwise, \end{cases} \tag{14}$$

$$a^{-1}(F^*) = a^{-1}F\cdot F^*. \tag{15}$$

Proposition 2 (Berry and Sethi [4]). *The derivative $u^{-1}E$ of a linear regular expression E w.r.t. a word u of Σ^+ is such that*

$$u^{-1}0 = 0, \tag{16}$$

$$u^{-1}1 = 0, \tag{17}$$

$$u^{-1}a = \begin{cases} 1 & if\ u = a, \\ 0 & otherwise, \end{cases} \tag{18}$$

$$u^{-1}(F+G) = \begin{cases} u^{-1}F & if\ u \in \Sigma_F^+, \\ u^{-1}G & if\ u \in \Sigma_G^+, \\ 0 & otherwise, \end{cases} \tag{19}$$

$$u^{-1}(F\cdot G)=\begin{cases} u^{-1}F\cdot G & \text{if } u\in\Sigma_F^+,\\ \sum_{u=ps,\,s\neq\varepsilon}\lambda(p^{-1}F)\cdot s^{-1}G & \text{otherwise,}\end{cases} \tag{20}$$

$$u^{-1}(F^*)=\sum_{u=ps,\,s\neq\varepsilon} v_s\cdot s^{-1}F\cdot F^* \quad \text{where } v_s\in\{0,1\}. \tag{21}$$

3.2. Continuations in a linear expression

For every symbol a of a linear expression E, the non-zero expressions $(ua)^{-1}E$, where u is any word of Σ^*, are called continuations of a in E. We denote by $C_a(E)$ the set of the continuations of a in E.

Theorem 2 (Berry and Sethi [4]). *For every symbol a of a linear expression E, the continuations of a in E are aci-similar.*

This result leads to the definition of the Berry–Sethi nondeterministic automaton $\mathscr{BS}_E$ of a regular expression E. Its set of states is $Pos_E\cup\{0\}$, where 0 is not in Pos_E. Its transitions are produced by the computation of the (sets of) continuations $C_x(\bar{E})$. This automaton recognizes $L(E)$. In the following we shall use the notation $C_0(\bar{E})=\{\varepsilon^{-1}\bar{E}\}=\{\bar{E}\}$.

Definition 4 (*Berry–Sethi automaton*). The Berry–Sethi automaton of a regular expression E, $\mathscr{BS}_E=(Q,\Sigma,i,T,\delta)$, is defined by

- $Q=Pos_E\cup\{0\}$,
- $i=0$,
- $T=\{x\mid C\in C_x(\bar{E})\Rightarrow\lambda(C)=1\}$,
- $\delta(x,a)=\{y\mid h(y)=a \text{ and } C\in C_x(\bar{E})\Rightarrow y^{-1}C\in C_y(\bar{E})\}$, $\forall x\in Q$ and $\forall a\in\Sigma$.

Notice the number of states of the automaton $\mathscr{B}_{\bar{E}}$ is less than or equal to the number of states of $\mathscr{BS}_{\bar{E}}$, since the continuations of a given symbol in $\bar{E}$ may be similar to E or/and to the continuations of other symbols.

Notice that the above definition involves *sets* of continuations $C_x(\bar{E})$. The reason is that Berry and Sethi do not associate a specific continuation to a state: they say that *the* continuation of a in E refers to *some* expression in the equivalence class. We make this point more explicit in the next section by actually computing canonical representatives of the continuation sets, which allows us to substitute identity tests to similarity tests.

3.3. Definition of c-derivatives

Definition 5 (*c-derivative w.r.t. a symbol*). Given a regular expression E and a symbol a, the c-derivative of E w.r.t. a, written d_aE, is defined by

$$d_a(0)=0, \tag{22}$$

$$d_a(1) = 0, \tag{23}$$

$$d_a(x) = \begin{cases} 1 & \text{if } a = x, \\ 0 & \text{otherwise,} \end{cases} \tag{24}$$

$$d_a(F + G) = \begin{cases} d_a(F) & \text{if } d_a(F) \neq 0, \\ d_a(G) & \text{otherwise,} \end{cases} \tag{25}$$

$$d_a(F \cdot G) = \begin{cases} d_a(F) \cdot G & \text{if } d_a(F) \neq 0, \\ \lambda(F) \cdot d_a(G) & \text{otherwise,} \end{cases} \tag{26}$$

$$d_a(F^*) = d_a(F) \cdot F^*. \tag{27}$$

Proposition 3. *Let E be a linear expression. Then for all $a \in \Sigma$, one has*

$$d_a(E) \equiv a^{-1}E.$$

Proof. The proof directly derives from Proposition 1. □

Definition 6 (*c-derivative w.r.t. a word*). C-derivatives w.r.t. a symbol can be extended to c-derivatives w.r.t. a word $u = u_1 \dots u_n$ as follows:

$$d_\varepsilon(E) = E, \tag{28}$$

$$d_{u_1 \dots u_n}(E) = d_{u_2 \dots u_n}(d_{u_1}(E)). \tag{29}$$

More generally, we shall use the equation

$$d_{ps}(E) = d_s(d_p(E)) \tag{30}$$

which holds for every factorization $u = ps$.

Proposition 4. *The c-derivative $d_u(E)$ of a linear expression E w.r.t. a word u of Σ^+ is either* 0 *or such that*

$$d_u(u) = 1, \tag{31}$$

$$d_u(F + G) = \begin{cases} d_u(F) & \text{if } d_u(F) \neq 0, \\ d_u(G) & \text{otherwise,} \end{cases} \tag{32}$$

$$d_u(F \cdot G) = \begin{cases} d_u(F) \cdot G & \text{if } d_u(F) \neq 0, \\ d_s(G) & \text{otherwise } (s \neq \varepsilon \text{ is some suffix of } u), \end{cases} \tag{33}$$

$$d_u(F^*) = d_s(F) \cdot F^* (s \neq \varepsilon \text{ is some suffix of } u). \tag{34}$$

Proof. The proof is by induction on the length of u and on the structure of E. The proposition is true for the c-derivative of every linear expression E w.r.t. every symbol a, by Proposition 3. We suppose that the proposition is satisfied for the c-derivative of

every linear expression w.r.t. every word u of length less than n, $n>1$, and we prove it is satisfied for words of length less than $n+1$.

Case $E=F+G$. By inductive hypothesis, we have

$$d_u(F+G)=\begin{cases} d_u(F) & \text{if } d_u(F)\neq 0,\\ d_u(G) & \text{otherwise.}\end{cases}$$

By definition (30), it implies

$$d_{ua}(F+G)=\begin{cases} d_{ua}(F) & \text{if } d_u(F)\neq 0,\\ d_{ua}(G) & \text{otherwise.}\end{cases}$$

Since $\Sigma_F\cap\Sigma_G=\emptyset$, it implies that $d_{ua}(F+G)$ is either 0 or such that:

$$d_{ua}(F+G)=\begin{cases} d_{ua}(F) & \text{if } d_{ua}(F)\neq 0,\\ d_{ua}(G) & \text{otherwise.}\end{cases}$$

Proof is similar in the case when $E=F\cdot G$ or $E=F^*$. □

Proposition 5. *Let E be a linear expression. Then for all $u\in\Sigma^*$, one has*

$$d_u(E)\sim_{\text{aci}} u^{-1}E.$$

Proof. It is easy to proof that $d_u(E)=0$ if and only if $u^{-1}E\sim_{\text{aci}}0$. In the following we assume that $d_u(E)\neq 0$. The proof is by induction on the structure of E. Let $u=u_1\dots u_n$. For the base cases where $|E|=1$, $d_u(E)$ and $u^{-1}E$ are simultaneously equal to 0 or 1. Thus the proposition is true for base cases. For the induction step, we must examine three cases and make use of Eqs. (16)–(21) and (31)–(34).

Case 1: $E=F+G$. If $d_u(F)\neq 0$, then we have $d_u(E)=d_u(F)$ and $d_u(G)=0$. From the inductive hypothesis $d_u(F)\sim_{\text{aci}}u^{-1}F$. Hence $d_u(E)\sim_{\text{aci}}u^{-1}E$. The proof is similar in the case when $d_u(F)=0$.

Case 2: $E=F\cdot G$. If $d_u(F)\neq 0$, then we have $d_u(E)=d_u(F)\cdot G$ and $u^{-1}E=u^{-1}F\cdot G$. From the inductive hypothesis, $d_u(F)\sim_{\text{aci}}u^{-1}F$. Hence $d_u(E)\sim_{\text{aci}}u^{-1}E$. If $d_u(F)=0$, then there exists a suffix s of u such that $d_u(E)=d_s(G)$. From the inductive hypothesis, $d_s(G)\sim_{\text{aci}}s^{-1}G$. It implies that $u^{-1}E$ is a non-zero sum: it contains $s^{-1}G$ and possibly expressions equivalent to $s^{-1}G$. Hence $d_u(E)\sim_{\text{aci}}u^{-1}E$.

Case 3: $E=F^*$. There exists a suffix s of u such that $d_u(E)=d_s(F)\cdot F^*$. From the inductive hypothesis, $d_s(F)\sim_{\text{aci}}s^{-1}F$. Thus $d_s(F)\cdot F^*\sim_{\text{aci}}s^{-1}F\cdot F^*$. It implies that $u^{-1}E$ is a non-zero sum: it contains $s^{-1}F\cdot F^*$ and possibly expressions equivalent to $s^{-1}F\cdot F^*$. Hence $d_u(E)\sim_{\text{aci}}u^{-1}E$. □

Example 1. Consider the linear expression $E=(a^*+b)^*$. In this example we can see that $d_a(E)=a^{-1}E$ and that $d_{aa}(E)\sim_{\text{aci}}(aa)^{-1}E$.

$$\begin{aligned} d_a((a^*+b)^*) &= d_a(a^*+b)(a^*+b)^* = d_a(a^*)(a^*+b)^*\\ &= d_a(a)a^*(a^*+b)^* = a^*(a^*+b)^*, \end{aligned}$$

$$a^{-1}(a^*+b)^* = (a^{-1}(a^*+b))(a^*+b)^* = (a^{-1}a^*)(a^*+b)^*$$
$$= (a^{-1}a)a^*(a^*+b)^* = a^*(a^*+b)^*,$$

$$d_{aa}((a^*+b)^*) = d_a(a^*(a^*+b)^*) = d_a(a^*)(a^*+b)^*$$
$$= d_a(a)a^*(a^*+b)^* = a^*(a^*+b)^*,$$

$$\begin{aligned}(aa)^{-1}(a^*+b)^* &= a^{-1}(a^*(a^*+b)^*)\\ &= (a^{-1}a^*)(a^*+b)^* + a^{-1}(a^*+b)^*\\ &= (a^{-1}a)a^*(a^*+b)^* + a^*(a^*+b)^*\\ &= a^*(a^*+b)^* + a^*(a^*+b)^*.\end{aligned}$$

Theorem 3. *Let E be a linear expression over an alphabet Σ. For all $u, v \in \Sigma^*$, one has*

$$u^{-1}E \sim_{\text{aci}} v^{-1}E \quad \Leftrightarrow \quad d_u(E) \equiv d_v(E).$$

Proof. ($\Leftarrow$) The proof is obvious.

($\Rightarrow$) The proof is by induction on the number of operators in E. Let us first consider the case of zero derivatives. Let u, v in Σ^* be such that $u^{-1}E \sim_{\text{aci}} 0$ and $u^{-1}E \sim_{\text{aci}} v^{-1}E$. It implies $v^{-1}E \sim_{\text{aci}} 0$ and $d_u(E) \equiv 0 \equiv d_v(E)$. Thus we shall assume now that $E \neq 0$ and we shall only consider cases when $u^{-1}E \neq 0$. The base cases are $E = 1$ and a for some a in Σ.

Case $E = 1$: Let u, v in Σ^* be such that $u^{-1}1 \sim_{\text{aci}} v^{-1}1$. By (17) $u^{-1}1 \neq 0$ and $v^{-1}1 \neq 0$ imply that $u = v = \varepsilon$ and $u^{-1}1 = v^{-1}1 = 1$. Since $u = v = \varepsilon$ implies $d_u(1) = d_v(1) = 1$, we get: $u^{-1}1 \sim_{\text{aci}} v^{-1}1 \Rightarrow d_u(1) \equiv d_v(1)$.

Case $E = a$, a in Σ: Let u, v in Σ^* be such that $u^{-1}a \sim_{\text{aci}} v^{-1}a$. By (18) it implies that either $u^{-1}a = v^{-1}a = a$ if $u = v = \varepsilon$, or $u^{-1}a = v^{-1}a = 1$ if $u = v = a$. By (31) we have $d_u(a) = d_v(a) = a$ if $u = v = \varepsilon$, or $d_u(a) = d_v(a) = 1$ if $u = v = a$. Thus $u^{-1}a \sim_{\text{aci}} v^{-1}a \Rightarrow d_u(a) \equiv d_v(a)$.

For the induction step there are three cases depending on the structure of E.

Case 1: $E = F + G$. Let u, v in Σ^* be such that $u^{-1}(F+G) \sim_{\text{aci}} v^{-1}(F+G)$. By Proposition 5, we have: $d_u(F+G) \sim_{\text{aci}} d_v(F+G)$. We consider four sub-cases: (a) $d_u(F) \neq 0$, $d_v(F) \neq 0$, (b) $d_u(F) \neq 0$, $d_v(F) = 0$, (c) $d_u(F) = 0$, $d_v(F) \neq 0$, and (d) $d_u(F) = 0$, $d_v(F) = 0$. We limit ourselves to the sub-cases (a) and (b); proof for (c) and (d) can be done in a similar way. In the sub-case (a), $d_u(F+G) = d_u(F)$ and $d_v(F+G) = d_v(F)$. By Proposition 5 we have $d_u(F) \sim_{\text{aci}} u^{-1}F$ and $d_v(F) \sim_{\text{aci}} v^{-1}F$. Since $d_u(F+G) \sim_{\text{aci}} d_v(F+G)$, we get $u^{-1}F \sim_{\text{aci}} v^{-1}F$ and by the inductive hypothesis $d_u(F) \equiv d_v(F)$.

In the sub-case (b), by a similar way we get that $u^{-1}F \sim_{\text{aci}} v^{-1}G$. Since $\Sigma_F \cap \Sigma_G = \emptyset$, $u^{-1}F \sim_{\text{aci}} v^{-1}G$ implies $u^{-1}F = 1 = v^{-1}G$. Consequently $d_u(F) = 1 = d_v(G)$.

Case 2: $E = F \cdot G$. Let u, v in Σ^* be such that $u^{-1}(F \cdot G) \sim_{\text{aci}} v^{-1}(F \cdot G)$. By Proposition 5 we have: $d_u(F \cdot G) \sim_{\text{aci}} d_v(F \cdot G)$.

We must consider the same four sub-cases as in Case 1. We limit ourself to the sub-cases (a) and (b); proofs for (c) and (d) can be done in a similar way.

In the sub-case (a), we have $d_u(F \cdot G) = d_u(F) \cdot G$ and $d_v(F \cdot G) = d_v(F) \cdot G$. Thus we get that $d_u(F) \cdot G \sim_{\text{aci}} d_v(F) \cdot G$, which implies by Proposition 5 that $u^{-1}F \cdot G \sim_{\text{aci}} v^{-1}F \cdot G$. Hence $u^{-1}F \sim_{\text{aci}} v^{-1}F$. Applying the inductive hypothesis on F we get $d_u(F) \equiv d_v(F)$. Thus $d_u(F) \cdot G \equiv d_v(F) \cdot G$.

In the sub-case (b), by a similar way we get that there exists a suffix s of v such that $u^{-1}F \cdot G \sim_{\text{aci}} s^{-1}G$. Since $\Sigma_F \cap \Sigma_G = \emptyset$, we have $u^{-1}F \sim_{\text{aci}} 1$, which implies that $d_u(F) \equiv 1$ and $s^{-1}G \sim_{\text{aci}} G$. Since we have $\varepsilon^{-1}G = G$, it comes $d_s(G) \equiv G$. Hence $d_u(F) \cdot G \equiv d_s(G)$.

Case 3: $E = F^*$. Using Proposition 5 and Eq. (34) we get $u^{-1}F^* \sim_{\text{aci}} d_u(F^*) = d_s(F) \cdot F^* \sim_{\text{aci}} s^{-1}F \cdot F^*$ and $v^{-1}F^* \sim_{\text{aci}} d_u(F^*) = d_t(F) \cdot F^* \sim_{\text{aci}} t^{-1}F \cdot F^*$ for some suffixes s of u and t of v. Then $u^{-1}F^* \sim_{\text{aci}} v^{-1}F^*$ is equivalent to $s^{-1}F \cdot F^* \sim_{\text{aci}} t^{-1}F \cdot F^*$. From the inductive hypothesis on F we get $d_s(F) \cdot F^* \sim_{\text{aci}} d_t(F) \cdot F^*$. Thus $d_u(F^*) \equiv d_v(F^*)$. □

Theorem 4. *If E is linear, for every symbol a and every word u, the c-derivative $d_{ua}(E)$ of E w.r.t. the word ua is either* 0 *or unique.*

Proof. This theorem is a straightforward consequence of Theorems 2 and 3. □

Theorem 4 allows us to define $c_a(E)$, the c-continuation of a in E, which is the unique value of the non-zero c-derivatives $d_{ua}(E)$. From Proposition 4, we get the following equations:

Proposition 6. *For every symbol a of a linear expression E, the c-continuation $c_a(E)$ is such that*

$$c_a(a) = 1, \tag{35}$$

$$c_a(F + G) = \begin{cases} c_a(F) & \text{if } c_a(F) \neq 0, \\ c_a(G) & \text{otherwise}, \end{cases} \tag{36}$$

$$c_a(F \cdot G) = \begin{cases} c_a(F) \cdot G & \text{if } c_a(F) \neq 0, \\ c_a(G) & \text{otherwise}, \end{cases} \tag{37}$$

$$c_a(F^*) = c_a(F) \cdot F^*. \tag{38}$$

Corollary 5. *For every symbol a of a linear expression E, the c-continuation $c_a(E)$ is either* 1 *or a subexpression of E or a product of subexpressions.*

3.4. Definition of the c-continuation automaton

Let E be a regular expression over Σ, $\bar{E}$ the linearized version of E over Pos_E and h the mapping from Pos_E onto Σ. We assume that 0 is a symbol not in Pos_E. Let $c_0(\bar{E}) = d_\varepsilon(\bar{E}) = \bar{E}$.

Theorem 4 leads to the definition of the non-deterministic automaton $\mathscr{C}_E$, called the c-continuation automaton of E. The states are pairs $(x, c_x(\bar{E}))$ with x in $Pos_E \cup \{0\}$. The transitions are produced by the computation of c-continuations.

Definition 7 (*c-continuation automaton*). The c-derivative automaton of E, $\mathscr{C}_E = (Q, \Sigma, i, T, \delta)$, is defined by

- $Q = \{(x, c_x(\bar{E})) \mid x \in Pos_E \cup \{0\}\}$,
- $i = (0, c_0(\bar{E}))$,
- $T = \{(x, c_x(\bar{E})) \mid \lambda(c_x(\bar{E})) = 1\}$,
- $\delta((x, c_x(\bar{E})), a) = \{(y, c_y(\bar{E})) \mid h(y) = a \text{ and } d_y(c_x(\bar{E})) \equiv c_y(\bar{E})\}$, $\forall x \in Pos_E \cup \{0\}$ and $\forall a \in \Sigma$.

Example 2. Consider the regular expression $E = x^*(xx + y)^*$. The linearized version of E is $\bar{E} = x_1^*(x_2x_3 + y_4)^*$. The states of the c-continuation automaton of E are

$$d_\varepsilon(\bar{E}) = \bar{E} = x_1^*(x_2x_3 + y_4)^* = c_0,$$

$$\begin{aligned} d_{x_1}(\bar{E}) &= d_{x_1}(x_1^*(x_2x_3 + y_4)^*) = d_{x_1}(x_1^*)(x_2x_3 + y_4)^* \\ &= d_{x_1}(x_1)x_1^*(x_2x_3 + y_4)^* = x_1^*(x_2x_3 + y_4)^* = c_{x_1}, \end{aligned}$$

$$\begin{aligned} d_{x_2}(\bar{E}) &= d_{x_2}(x_1^*(x_2x_3 + y_4)^*) = d_{x_2}((x_2x_3 + y_4)^*) \\ &= d_{x_2}(x_2x_3 + y_4)(x_2x_3 + y_4)^* = d_{x_2}(x_2x_3)(x_2x_3 + y_4)^* \\ &= x_3(x_2x_3 + y_4)^* = c_{x_2}, \end{aligned}$$

$$d_{x_3}(\bar{E}) = 0,$$

$$\begin{aligned} d_{y_4}(\bar{E}) &= d_{y_4}(x_1^*(x_2x_3 + y_4)^*) = d_{y_4}((x_2x_3 + y_4)^*) \\ &= d_{y_4}(x_2x_3 + y_4)(x_2x_3 + y_4)^* = d_{y_4}(y_4)(x_2x_3 + y_4)^* \\ &= (x_2x_3 + y_4)^* = c_{x_4}, \end{aligned}$$

$$\begin{aligned} d_{x_2x_3}(\bar{E}) &= d_{x_3}(x_3(x_2x_3 + y_4)^*) = d_{x_3}(x_3)(x_2x_3 + y_4)^* \\ &= (x_2x_3 + y_4)^* = c_{x_3}. \end{aligned}$$

We can produce the transitions in a similar way, and we finally get the following automaton (Fig. 1):

The two next sections enlighten the interest of the $\mathscr{C}_E$ automaton: first it coincides with the position automaton of E as far as its states are viewed as positions, and

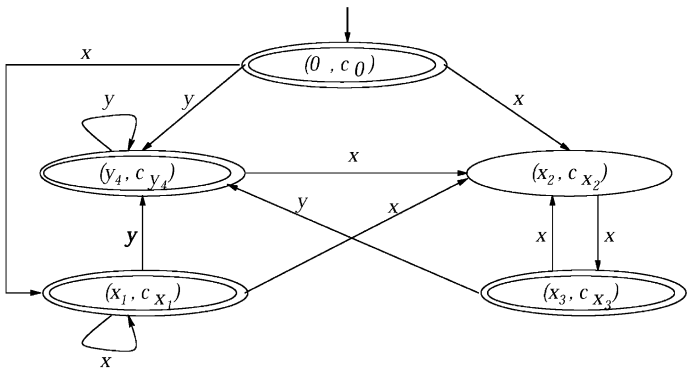

Fig. 1. The c-continuation automaton of $x^*(xx+y)^*$.

secondly it can be factored into the equation automaton if its states are viewed as c-continuations.

4. Construction of the position automaton

Let E be a regular expression over Σ, Pos_E the set of its positions, $\bar{E}$ its linearized version over Pos_E, and h the mapping from Pos_E onto Σ. Let us consider the following positions sets:

- $First(E)$, the set of positions that match the first symbol of some word in $L(\bar{E})$.
- $Last(E)$, the set of positions that match the last symbol of some word in $L(\bar{E})$.
- $Follow(E,x)$, for all x in Pos_E: the set of positions that follow the position x in some word of $L(\bar{E})$.

The position automaton $\mathscr{P}_E$ of the regular expression E is derived from the above position sets as follows.

Definition 8 (*position automaton*). The position automaton of E, $\mathscr{P}_E=(Q_P,\Sigma,i_P,T_P,\delta_P)$, is defined by

- $Q_P=Pos_E\cup\{0\}$,
- $i_P=0$,
- $T_P=\begin{cases} Last(E) & \text{if } \lambda(E)=0,\\ Last(E)\cup\{0\} & \text{otherwise,}\end{cases}$
- $\delta_P(0,a)=\{x\in First(E)\mid h(x)=a\},\ \forall a\in\Sigma$,
- $\delta_P(x,a)=\{y\mid y\in Follow(E,x) \text{ and } h(y)=a\},\ \forall x\in Pos_E,\ \forall a\in\Sigma$.

Example 3. Consider the regular expression $E=x^*(xx+y)^*$. The linearized version of E is $\bar{E}=x_1^*(x_2x_3+y_4)^*$. We have

$First(E)=\{x_1,x_2,y_4\}$, $Last(E)=\{x_1,x_3,y_4\}$, $\lambda(E)=1$ and $Follow(E,x_1)=\{x_1,x_2,y_4\}$, $Follow(E,x_2)=\{x_3\}$, $Follow(E,x_3)=Follow(E,y_4)=\{x_2,y_4\}$ (Fig. 2).

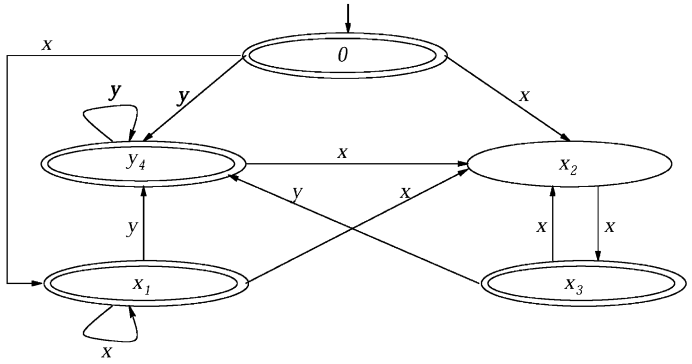

Fig. 2. The position automaton of $x^*(xx+y)^*$.

Proposition 7. *The automaton $\mathscr{P}_E$ recognizes the language $L(E)$.*

Proof is given in [8, 11] where the well-known construction of the position automaton based on a recursive computation of the sets *First*, *Last* and *Follow* appeared first and independently.

We now show that the computation of the c-continuations leads to a construction of the position automaton. We consider the automaton $\mathscr{C}_E=(Q,\Sigma,i,T,\delta)$. Let $p:Q\to Pos_E\cup\{0\}$ be such that $p(x,c_x(\bar{E}))=x$. The automaton $p(\mathscr{C}_E)$ is obtained from $\mathscr{C}_E$ by replacing Q by $Pos_E\cup\{0\}$.

Theorem 6. *Let E be a regular expression. The automaton $p(\mathscr{C}_E)$, based on the c-continuations of $\bar{E}$, and $\mathscr{P}_E$, the position automaton, are identical.*

Proof. The automata $p(\mathscr{C}_E)$ and $\mathscr{P}_E$ have identical sets of states and identical initial states. As for final states and transitions, identity is a direct consequence of the following proposition. □

Proposition 8. *Let E be a regular expression. Then the following equalities hold*:
(1) $First(E)=\{y\in Pos_E \mid d_y(\bar{E})\neq 0\}$;
(2) $Last(E)=\{y\in Pos_E \mid \lambda(c_y(\bar{E}))=1\}$;
(3) $Follow(E,x)=\{y\in Pos_E \mid d_y(c_x(\bar{E}))\neq 0\}$.

Proof. Let u and v be words in Pos_E^*. From the definition of sets *First*, *Last* and *Follow*, and from Proposition 5, we obtain

$$\begin{aligned}First(E)&=\{y\in Pos_E \mid yv\in L(\bar{E})\}=\{y\in Pos_E \mid v\in y^{-1}L(\bar{E})\}\\&=\{y\in Pos_E \mid v\in L(y^{-1}\bar{E})\}=\{y\in Pos_E \mid L(y^{-1}\bar{E})\neq\emptyset\}\\&=\{y\in Pos_E \mid L(d_y(\bar{E}))\neq\emptyset\}=\{y\in Pos_E \mid d_y(\bar{E})\neq 0\},\end{aligned}$$

$$\begin{aligned} Last(E) &= \{y \in Pos_E \mid vy \in L(\bar{E})\} = \{y \in Pos_E \mid \varepsilon \in (vy)^{-1}L(\bar{E})\} \\ &= \{y \in Pos_E \mid \varepsilon \in L((vy)^{-1}\bar{E})\} = \{y \in Pos_E \mid \varepsilon \in L(d_{vy}(\bar{E}))\} \\ &= \{y \in Pos_E \mid \lambda(d_{vy}(\bar{E})) = 1\} = \{y \in Pos_E \mid \lambda(c_y(\bar{E})) = 1\}, \end{aligned}$$

$$\begin{aligned} Follow(E,x) &= \{y \in Pos_E \mid uxyv \in L(\bar{E})\} \\ &= \{y \in Pos_E \mid (uxy)^{-1}L(\bar{E}) \neq \emptyset\} \\ &= \{y \in Pos_E \mid L((uxy)^{-1}\bar{E}) \neq \emptyset\} \\ &= \{y \in Pos_E \mid L(d_{uxy}(\bar{E})) \neq \emptyset\} \\ &= \{y \in Pos_E \mid d_{uxy}(\bar{E}) \neq 0\} \\ &= \{y \in Pos_E \mid d_y(d_{ux}\bar{E}) \neq 0\} \\ &= \{y \in Pos_E \mid d_y(c_x(\bar{E})) \neq 0\}. \end{aligned}$$

Corollary 7. *The automaton $\mathscr{C}_E$ recognizes the language $L(E)$.*

Notice that this result is very close to Berry–Sethi's one. The notion of c-derivative is a nice tool to present it. Next results will enlighten the interest of this notion.

5. From c-derivatives to partial derivatives

We first recall the definition and the properties of partial derivatives of a regular expression and then compare them to c-derivatives.

5.1. Partial derivatives

Partial derivatives of regular expressions have been introduced by Antimirov in [2]. They are, in a sense, a "non-deterministic generalization" of word derivatives: every derivative of E can be represented by a finite set of some partial derivatives of E. For example, for $E = ab + a^*$ and $a^{-1}E = b + a^*$, b and a^* are the two partial derivatives of E w.r.t. a.

Let us mention that the notion of partial derivative is very close to the notion of prebase due to Mirkin [12]. A full comparison of these two concepts is given in [7].

Definition 9 (*set of partial derivatives w.r.t. a symbol*). Given a regular expression E and a symbol a, the set of partial derivatives of E w.r.t. a, written $\partial_a(E)$, is recursively defined on the structure of E as follows:

$$\partial_a(0) = \emptyset, \tag{39}$$

$$\partial_a(1) = \emptyset, \tag{40}$$

$$\partial_a(x) = \{1\} \quad \text{if } a = x \quad \emptyset \text{ otherwise}, \tag{41}$$

$$\partial_a(F + G) = \partial_a(F) \cup \partial_a(G), \tag{42}$$

$$\partial_a(F \cdot G) = \begin{cases} \partial_a(F) \cdot G & \text{if } \lambda(F) = 0, \\ \partial_a(F) \cdot G \cup \partial_a(G) & \text{otherwise}, \end{cases} \tag{43}$$

$$\partial_a(F^*) = \partial_a(F) \cdot F^*. \tag{44}$$

Definition 10 (*extensions*). The symbol a in $\partial_a(E)$ can be replaced by any word u of Σ^* or by any set of words U, and the expression E can be replaced by any set R of expressions, according to the equations:

$$\partial_\varepsilon(E) = \{E\}, \tag{45}$$

$$\partial_{ua}(E) = \partial_a(\partial_u(E)), \tag{46}$$

$$\partial_u(R) = \bigcup_{E \in R} \partial_u(E), \tag{47}$$

$$\partial_U(E) = \bigcup_{u \in U} \partial_u(E). \tag{48}$$

Proposition 9 (Antimirov [2]). *The set $\partial_{ua}(E)$ of partial derivatives of a regular expression E w.r.t. a word ua of Σ^+ is such that*

$$\partial_{ua}(0) = \emptyset, \tag{49}$$

$$\partial_{ua}(1) = \emptyset, \tag{50}$$

$$\partial_{ua}(x) = \begin{cases} \{1\} & \textit{if } u = \varepsilon \textit{ and } x = a, \\ \emptyset & \textit{otherwise}, \end{cases} \tag{51}$$

$$\partial_{ua}(F + G) = \partial_{ua}(F) \cup \partial_{ua}(G), \tag{52}$$

$$\partial_{ua}(F \cdot G) \subseteq \partial_{ua}(F) \cdot G \cup \bigcup_{ua = psa} \partial_{sa}(G), \tag{53}$$

$$\partial_{ua}(F^*) \subseteq \bigcup_{ua = psa} \partial_{sa}(F) \cdot F^*. \tag{54}$$

Let $\mathscr{P}D(E) = \partial_{\Sigma^*}(E)$ be the set of all partial derivatives of the regular expression E.

Theorem 8 (Antimirov [2]). *The cardinality of the set $\mathscr{P}D(E)$ of all partial derivatives of a regular expression E is less than or equal to $\|E\| + 1$.*

Hence the definition of the equation automaton $\mathscr{E}_E$ of E, whose set of states is the set of all partial derivatives of E, and which recognizes $L(E)$.

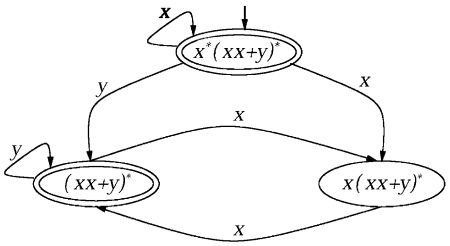

Fig. 3. The equation automaton of $x^*(xx+y)^*$.

Definition 11 (*equation automaton*). The equation automaton of a regular expression E, $\mathscr{E}_E = (Q, \Sigma, i, T, \delta)$, is defined by

- $Q = \mathscr{P}D(E)$,
- $i = E$,
- $T = \{p \mid \lambda(p) = 1\}$,
- $\delta(p,a) = \partial_a(p)$, $\forall p \in Q$ and $\forall a \in \Sigma$.

Example 4. In this example we show the computation of states and transitions of the equation automaton of $E = x^*(xx+y)^*$ (Fig. 3).

$$\partial_\varepsilon(E) = E = x^*(xx+y)^*,$$

$$\begin{aligned}\partial_x(E) &= \partial_x(x^*)(xx+y)^* \cup \partial_x((xx+y)^*)\\ &= \partial_x(x)x^*(xx+y)^* \cup (\partial_x(xx) \cup \partial_x(y))(xx+y)^*\\ &= \{x^*(xx+y)^*\} \cup \{x(xx+y)^*\}\\ &= \{x^*(xx+y)^*,\ x(xx+y)^*\},\end{aligned}$$

$$\begin{aligned}\partial_y(E) &= \partial_y(x^*)(xx+y)^* \cup \partial_y((xx+y)^*)\\ &= \partial_y(x)x^*(xx+y)^* \cup (\partial_y(xx) \cup \partial_y(y))(xx+y)^*\\ &= \emptyset \cup (\emptyset \cup \{1\})(xx+y)^*\\ &= \{(xx+y)^*\},\end{aligned}$$

$$\partial_x(x(xx+y)^*) = \partial_x(x)(xx+y)^* = \{(xx+y)^*\},$$

$$\partial_y(x(xx+y)^*) = \partial_y(x)(xx+y)^* = \emptyset,$$

$$\begin{aligned}\partial_x((xx+y)^*) &= \partial_x(xx+y)(xx+y)^* = (\partial_x(xx) \cup \partial_x(y))(xx+y)^*\\ &= \{x(xx+y)^*\},\end{aligned}$$

$$\begin{aligned}\partial_y((xx+y)^*) &= \partial_y(xx+y)(xx+y)^* = (\partial_y(xx) \cup \partial_y(y))(xx+y)^*\\ &= \{(xx+y)^*\}.\end{aligned}$$

5.2. C-derivatives versus partial derivatives

We now explain how c-derivatives and partial derivatives w.r.t. a symbol are related.

Proposition 10. *Let E be a regular expression. Let H_E be a subexpression of E. Denote by $P(a_i, H_E)$ the property*: $(a_i \in Pos_E(H_E)) \wedge (h(a_i) = a) \wedge (d_{a_i}(\overline{H_E}) \neq 0)$. *Then for all H_E one has*

$$\bigcup_{P(a_i,H_E)} h(d_{a_i}(\overline{H_E})) = \partial_a(H_E).$$

Proof. Subexpressions H_E, F_E and G_E are denoted by H, F and G for short, and $\overline{H_E}, \overline{F_E}$ and $\overline{G_E}$ are denoted by $\bar{H}$, $\bar{F}$ and $\bar{G}$. Let us verify that the proposition is true for atomic expressions. If $H = 0$ or $H = 1$, then we have $d_{a_i}(\bar{H}) = 0$ and $\partial_a(H) = \emptyset$. If $H = a$, then we have $\bar{H} = a_i$. Thus $h(d_{a_i}(\bar{H})) = 1$ and $\partial_a(E) = \{1\}$.

We now suppose that the proposition is true for every expression E of size less than n, $n > 1$, and we prove that it is true for expressions of size equal to n.

Case 1: $H = F + G$. We have $\overline{F + G} = \bar{F} + \bar{G}$. From Definition 5, $d_{a_i}(\bar{F} + \bar{G})$ is either equal to $d_{a_i}(\bar{F})$ or to $d_{a_i}(\bar{G})$. Thus, we can write

$$\begin{aligned}\bigcup_{P(a_i,H)} h(d_{a_i}(\bar{H})) &= \left(\bigcup_{P(a_i,F)} h(d_{a_i}(\bar{F}))\right) \cup \bigcup_{P(a_i,G)} h(d_{a_i}(\bar{G})) \\ &\overset{\text{ind.hyp.}}{=} \partial_a(F) \cup \partial_a(G) = \partial_a(H).\end{aligned}$$

Case 2: $H = F \cdot G$. We have $\overline{F \cdot G} = \bar{F} \cdot \bar{G}$. From Definition 5 there are two alternatives. The first one is when $d_{a_i}(\bar{F}) \neq 0$. In this case we have: $d_{a_i}(\bar{F} \cdot \bar{G}) = d_{a_i}(\bar{F}) \cdot \bar{G}$ and $a_i \in Pos_E(F)$. Thus we get

$$\begin{aligned}\bigcup_{P(a_i,H)} h(d_{a_i}(\bar{H})) &= \bigcup_{P(a_i,F)} h(d_{a_i}(\bar{F}) \cdot \bar{G}) = \left(\bigcup_{P(a_i,F)} h(d_{a_i}(\bar{F}))\right) \cdot G \\ &\overset{\text{ind.hyp.}}{=} \partial_a(F) \cdot G = \partial_a(H).\end{aligned}$$

In the second alternative we have: $\lambda(\bar{F}) = 1$, $d_{a_i}(\bar{F} \cdot \bar{G}) = d_{a_i}(\bar{G})$, $P(a_i, H) \Rightarrow P(a_i, G)$ and $a_i \in Pos_E(G)$. Hence, we can write

$$\bigcup_{P(a_i,H)} h(d_{a_i}(\bar{H})) = \bigcup_{P(a_i,G)} h(d_{a_i}(\bar{G})) \overset{\text{ind.hyp.}}{=} \partial_a(G) = \partial_a(H).$$

Case 3: $H = F^*$. We have $\overline{F^*} = \bar{F}^*$.

$$\begin{aligned}\bigcup_{P(a_i,H)} h(d_{a_i}(\bar{H})) &= \bigcup_{P(a_i,F)} h(d_{a_i}(\bar{F}) \cdot \bar{F}^*) = \left(\bigcup_{P(a_i,F)} h(d_{a_i}(\bar{F}))\right) \cdot F^* \\ &\overset{\text{ind.hyp.}}{=} \partial_a(F) \cdot F^* = \partial_a(H). \qquad \square\end{aligned}$$

We now deal with c-derivatives and partial derivatives w.r.t. a word. We first prove two propositions and then state a main theorem.

Proposition 11. *Let $d_u(\bar{E}) = H_1 \cdot H_2 \cdots H_l$ be a c-derivative. Let i and j be such that $1 \leqslant i < j \leqslant l$ and $Pos_E(H_i) \cap Pos_E(H_j) \neq \emptyset$. Then there exists a suffix s of u such that $d_u(\bar{E}) = d_s(H_j) \cdot H_{j+1} \cdots H_l$.*

Proof. From Proposition 4, if $d_u(\bar{E}) = H_1 \cdot H_2 \cdots H_l$, there exist a suffix s of u and a subexpression H' of $\bar{E}$ such that $d_u(\bar{E}) = d_s(H') \cdot H_j \cdot H_{j+1} \cdots H_l$, for some j, $1 \leqslant j \leqslant l$. If $\Sigma_{d_s(H')} \cap \Sigma_{H_j} \neq \emptyset$, then the product $d_s(H') \cdot H_j$ comes from the c-derivation of some star expression. That is $H_j = F^*$ and $H' = F$. Thus $d_u(\bar{E}) = d_s(F) \cdot F^* \cdot H_{j+1} \cdots H_l = d_s(F^*) \cdot H_{j+1} \cdots H_l = d_s(H_j) \cdot H_{j+1} \cdots H_l$. □

Let E be a regular expression and $F = d_u(\bar{E})$ a non-zero c-derivative of $\bar{E}$. We consider the linearization $\overline{h(F)}$ of $h(F)$ and we denote by h' the induced mapping from $Pos_{h(F)}$ onto Σ_E. For example if $E = (ab + b)^*$ and $F = d_{a_1}(\bar{E}) = b_2(a_1 b_2 + b_3)^*$, we have: $\overline{h(F)} = \overline{b(ab + b)^*} = b_1(a_2 b_3 + b_4)^*$.

Since $F = H_1 \cdot H_2 \cdots H_l$, where H_i is a linear subexpression of $\bar{E}$, for $1 \leqslant i \leqslant l$, we have $\overline{h(F)} = H'_1 \cdot H'_2 \cdots H'_l$, where H_i and H'_i, $1 \leqslant i \leqslant l$, are two linearizations of the same expression. In our current example, we have: $H_2 = (a_1 b_2 + b_3)^*$ and $H'_2 = (a_2 b_3 + b_4)^*$.

Let a_j be the jth symbol of $\overline{h(F)}$, and $\mu(a_j)$ the jth symbol of F. Notice that two distinct symbols of $\overline{h(F)}$ may be mapped to the same symbol of F: for instance, b_1 and b_3 are mapped to b_2.

The following proposition shows that it is equivalent to compute the c-derivative of F w.r.t. a_i and the c-derivative of $\overline{h(F)}$ w.r.t. every a_j such that $\mu(a_j) = a_i$.

Proposition 12. *Let E be a regular expression, $F = d_u(\bar{E})$ a non-zero c-derivative of $\bar{E}$, and h' the mapping associated to $\overline{h(F)}$. Let a_i a position of E, and a_j a position of $h(F)$ such that: $\mu(a_j) = a_i$, $h'(a_j) = h(a_i)$ and $d_{a_j}(\overline{h(F)}) \neq 0$. Then there exists m, $1 \leqslant m \leqslant l$, such that*

$$d_{a_j}(\overline{h(F)}) = d_{a_j}(H'_m) \cdot H'_{m+1} \cdots H'_l, \tag{55}$$

$$d_{a_i}(F) = d_{a_i}(H_m) \cdot H_{m+1} \cdots H_l, \tag{56}$$

$$h'(d_{a_j}(\overline{h(F)})) = h(d_{a_i}(F)). \tag{57}$$

Proof. We consider two cases. The first case is when there is no occurrence of a_i in F before rank j. We suppose that a_i occurs in H_m, which implies: $d_{a_i}(H_1) = d_{a_i}(H_2) = \cdots = d_{a_i}(H_{m-1}) = 0$. Since $d_{a_j}(\overline{h(F)}) \neq 0$ and a_j (only) occurs in H'_m, we have: $\lambda(H'_1) = \lambda(H'_2) = \cdots = \lambda(H'_{m-1}) = 1$, $d_{a_j}(H'_m) \neq 0$ and $d_{a_j}(\overline{h(F)}) = d_{a_j}(H'_m) \cdot H'_{m+1} \cdots H'_l$. As a consequence we have: $\lambda(H_1) = \lambda(H_2) = \cdots = \lambda(H_{m-1}) = 1$, $d_{a_i}(H_m) \neq 0$ and $d_{a_i}(F) = d_{a_i}(H_m) \cdot H_{m+1} \cdots H_l$. Therefore we get: $h'(d_{a_j}(\overline{h(F)})) = h(d_{a_i}(F))$.

The second case is when a_i occurs in F before rank j. In this case, by Proposition 11 there exists a suffix s of u such that $d_u(\bar{E}) = d_s(H_m)\cdot H_{m+1}\cdots H_l$. It implies that $d_{a_i}(F) = d_{a_i}(d_s(H_m)\cdot H_{m+1}\cdots H_l)$. Since $d_{a_j}(H'_m) \neq 0$ we have $d_{a_i}(H_m) \neq 0$. By Proposition 11, it comes $d_{sa_i}(H_m) = d_{a_i}(H_m)$. Finally we get: $d_{a_i}(F) = d_{a_i}(H_m)\cdot H_{m+1}\cdots H_l$ and $h'(d_{a_j}(\overline{h(F)})) = h(d_{a_i}(F))$. □

The following theorem enlightens the relation between c-derivatives and partial derivatives.

Theorem 9. *Let E be a regular expression and H_E be a subexpression of E. Let $P(u,H_E)$ denote the property: $(v = \bar{u} \in Pos^*_E(H_E)) \wedge (h(v) = u) \wedge (d_v(\overline{H_E}) \neq 0)$. Then for all subexpression H_E of E, one has*

$$\bigcup_{P(u,H_E)} h(d_{\bar{u}}(\overline{H_E})) = \partial_u(H_E).$$

Proof. The proof is done by induction on the length of u. The proposition is true for words $u = a$ by Proposition 10.

Assume now that the proposition is true for words u of length less than some integer n, $n > 1$, and prove it for words ua of length n. Denote by F the c-derivative $d_{\bar{u}}(\overline{H_E})$. By Proposition 12, we have

$$\bigcup_{P(ua,H_E)} h(d_{\bar{u}a_i}(\overline{H_E})) = \bigcup_{P(u,H_E),P(a_j,h(F))} h'(d_{a_j}(\overline{h(F)})).$$

By Proposition 10, we get

$$\bigcup_{P(u,H_E),P(a_j,h(F))} h'(d_{a_j}(\overline{h(F)})) = \partial_a\left(\bigcup_{P(u,H_E)} h(F)\right) \overset{\text{ind.hyp.}}{=} \partial_a(\partial_u(H_E)). \quad \square$$

6. Construction of the equation automaton

Let E be a regular expression over Σ and $\bar{E}$ the linearized version of E over Pos_E. We now present the connection between the two automata: (1) $\mathscr{C}_E$, the c-continuation automaton, and (2) $\mathscr{E}_E$, the equation automaton.

Let us first recall the definitions of $\mathscr{C}_E$ and $\mathscr{E}_E$ (from now on, $c_x(\bar{E})$ is shortened to c_x and h_E to h).

$\mathscr{C}_E = (Q, \Sigma, i, T, \delta)$, with $Q = \{(x, c_x) \mid x \in Pos_E \cup \{0\}\}$, $i = (0, c_0)$, $T = \{(x, c_x) \mid \lambda(c_x) = 1\}$, $\delta((x, c_x), a) = \{(y, c_y) \mid h(y) = a \text{ and } d_y(c_x) \equiv c_y\}$, $\forall x \in Pos_E \cup \{0\}$ and $\forall a \in \Sigma$.

$\mathscr{E}_E = (Q_A, \Sigma, i_A, T_A, \delta_A)$, with $Q_A = \mathscr{PD}(E)$, $i_A = E$, $T_A = \{p \mid \lambda(p) = 1\}$, $\delta_A(p, a) = \partial_a(p)$, $\forall p \in Q_A$ and $\forall a \in \Sigma$.

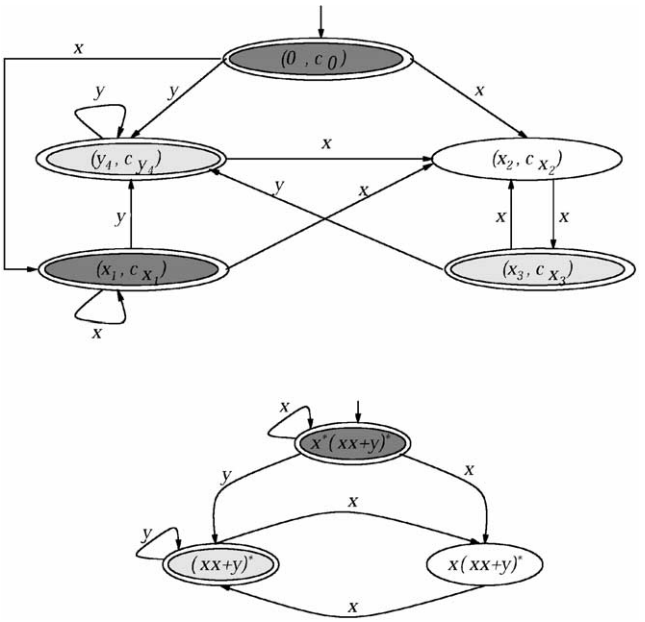

Fig. 4. The automata $\mathscr{C}_E$ and $\mathscr{C}_E/\sim$ for $E = x^*(xx + y)^*$.

Let us consider the following equivalence relation on $Pos_E \cup \{0\}$, the set of states of $\mathscr{C}_E$:

$$(x, c_x) \sim (y, c_y) \quad \Leftrightarrow \quad h(c_x) \equiv h(c_y).$$

Proposition 13. *The relation $\sim$ is right-invariant, i.e. for all a in Σ, for all (x, c_x), (t, c_t) in Q such that $(x, c_x) \sim (t, c_t)$, we have: $\delta((x, c_x), a) \sim \delta((t, c_t), a)$.*

Proof. Let $(x, c_x) \sim (t, c_t)$. We show that for every symbol a in Σ, for every transition $((x, c_x), a, (y, c_y))$ there exists a transition $((t, c_t), a, (z, c_z))$ such that $(y, c_y) \sim (z, c_z)$.

Assume that $(y, c_y) \in \delta((x, c_x), a)$. By definition of δ we have: $h(y) = a$ and $d_y(c_x) \equiv c_y$. Then, by Proposition 10, it comes $h(c_y) \in \partial_a(h(c_x))$. This implies that $h(c_y) \in \partial_a(h(c_t))$, since $h(c_x) \equiv h(c_t)$. By Proposition 10, there exists $z \in Pos_E$ such that (1) $h(z) = a$, which implies $(z, c_z) \in \delta((t, c_t), a)$ and (2) $h(d_z(c_t)) \equiv h(c_y)$, which implies $h(c_z) \equiv h(c_y)$ and thus $(y, c_y) \sim (z, c_z)$. □

Moreover, if two states are equivalent, then they are either both final or both non-final, since $(x, c_x) \in T \Leftrightarrow \lambda(c_x) = 1 \Leftrightarrow \lambda(h(c_x)) = 1$.

Let us denote by $[c_x]$ the equivalence class of the state (x, c_x). Since the relation $\sim$ is right-invariant we can define the quotient automaton $\mathscr{C}_E/\sim = (Q_\sim, \Sigma, i, T, \delta)$ as follows (Fig. 4):

- $Q_\sim = \{[c_x] \mid x \in Pos_E \cup \{0\}\}$,
- $i = [c_0]$,

- $T = \{[c_x] \mid \lambda(c_x) = 1\}$,
- $\delta([c_x], a) = \{[c_y] \mid h(y) = a \text{ and } d_y(c_x) \equiv c_y\}$, $\forall [c_x] \in Q_\sim$ and $\forall a \in \Sigma$.

Theorem 10. *Let E be a regular expression. The automaton $\mathscr{C}_E/_\sim$ derived from the c-continuation automaton is isomorphic to the equation automaton.*

Proof. By Theorem 9 the mapping h from Pos_E onto Σ is a surjective one from $\{c_x \mid x \in Pos_E\}$ onto $\partial_{\Sigma^+}(E)$. Since $c_0 = \bar{E}$ and $\partial_\varepsilon(E) = \{E\}$, we have $h(c_0) \in \partial_\varepsilon(E)$. Thus h is a surjective mapping from $Q = \{(x, c_x) \mid x \in Pos_E \cup \{0\}\}$ onto $Q_A = \mathscr{PD}(E)$. Thus, by definition of the equivalence $\sim$, h is a one-to-one mapping from $Q_\sim$ onto Q_A.

It can be easily checked that, by Proposition 10, we have: (1) $h(i) = h([\bar{E}]) = E = i_A$; (2) $h([T]) = T_A$; (3) h is compatible with transition functions: $[c_y] \in \delta([c_x], a) \Leftrightarrow h([c_y]) \in \delta_A(h([c_x]), a)$. □

7. An algorithm to compute the automaton $\mathscr{C}_E I_\sim$

We now present the sketch of the algorithm *CtoE*3 which computes the automaton $\mathscr{C}_E/_\sim$. We shortly describe a naive implementation of this algorithm, involving a complete computation of the c-continuations of the expression $\bar{E}$, and leading to an $O(||E|| \cdot |E|^2)$ space and time complexity. A full proof of the correctness and complexity analysis of this algorithm as well as algorithmic refinements yielding an $O(|E|^2)$ space and time complexity are presented in [6].

Let us notice that the alphabetic width and the size of a regular expression are *a priori* independent parameters. Indeed, for a given language, the size of an expression can be arbitrarily increased by adding 0's, concatenating 1's, or repetitively starring star subexpressions, with a constant alphabetic width. Moreover, some computation steps depend on the size of the expression while other ones depend on the alphabetic width. This additional information may be helpful, for implementation purpose for instance, or for a deeper analysis of the number of operations. Therefore complexities are expressed w.r.t. both of these two parameters.

7.1. Sketch of the algorithm

The algorithm *CtoE*3 is divided into three parts.

Part 1 is the computation of the set of states $Q_\sim$. The two following steps are performed:

(1) The production of the c-continuations of $\bar{E}$.
    By Corollary 5, a c-continuation can be computed as a product of subexpressions of $\bar{E}$. There are $O(|E|)$ subexpressions, and a subexpression has an $O(|E|)$ size. This implies that the size of a c-continuation is $O(|E|^2)$. The production of a c-continuation can be achieved on the syntax tree of E in space and time linear w.r.t.

the size of the c-continuation. Hence the whole step space and time complexity is $O(||E|| \cdot |E|^2)$.

(2) The identification of the c-continuations which coincide on Σ.
This step can be achieved by sorting the list of expressions in lexicographic order, according to Aho et al. algorithm [1] or Paige and Tarjan refinement [13], which lead to a time linear w.r.t. the sum of the size of the strings. Hence an $O(||E|| \cdot |E|^2)$ time complexity for this step.

Finally, the set of states of $\mathscr{C}_E/_\sim$ is computed with an $O(||E|| \cdot |E|^2)$ space and time complexity.

Part 2 is the computation of T, the set of final states. A final state of $\mathscr{C}_E/_\sim$ is a class of c-continuations c such that $\lambda(c) = 1$. Hence the two following steps:

(1) The computation of $\lambda(F)$ for each subexpression F of E.
This step can be achieved by a recursive procedure with an $O(|E|)$ space and time complexity.

(2) The computation of $\lambda(c)$ for one c-continuation c in each class.
Since a c-continuation is a product of $O(|E|)$ subexpressions of $\bar{E}$, $\lambda(c)$ can be computed with an $O(|E|)$ time complexity. Hence an $O(||E||)$ space complexity and an $O(||E|| \cdot |E|)$ time complexity for this step.

Finally, the set of final states of $\mathscr{C}_E/_\sim$ is computed with an $O(||E|| + |E|)$ space complexity and an $O(||E|| \cdot |E|)$ time complexity.

Part 3 is the computation of δ, the set of transitions. A transition $([c_x], a, [c_y])$ is such that $h(y) = a$ and $d_y(c_x) \equiv c_y$. On the other hand, we have: $\{y \in Pos_E \mid d_y(c_x) \neq 0\} = First(c_x)$. Hence the computation of δ is based on two steps:

(1) The computation of $First(\overline{F_E})$ for each subexpression $\overline{F_E}$ of $\bar{E}$.
This step can be achieved by a recursive procedure with an $O(||E|| \cdot |E|)$ space and time complexity.

(2) The computation of $First(c)$ for one c-continuation c in each class.
Since a c-continuation is a product of $O(|E|)$ subexpressions of $\bar{E}$, and since the First set of a subexpression has an $O(||E||)$ size, $First(c)$ can be computed with an $O(||E|| \cdot |E|)$ time complexity. Hence an $O(||E||^2)$ space complexity and an $O(||E||^2|E|)$ time complexity for this step.

Finally, the computation of the set of transitions of $\mathscr{C}_E/_\sim$ has an $O(||E|| \cdot |E|)$ space complexity and an $O(||E||^2|E|)$ time complexity.

The above results lead to the following theorem.

Theorem 11. *The Algorithm CtoE*3 *computes the* $\mathscr{C}_E/_\sim$ *automaton of a regular expression* E *with an* $O(||E|| \cdot |E|^2)$ *space and time complexity.*

Notice that the Algorithm *CtoE*3 provides an $O(||E|| \cdot |E|^2)$ construction of both the position automaton and the equation automaton of an expression.

7.2. Example

The following results are output by the program *ProgCtoE*3 which has been run on an example given by Antimirov in [2].

```
E=(a+b)*.(b.a.b.a.b.(a+b)*.b.a.b+b.b.a.(a+b)*.b.a.b).(a+b)*
size=47 alphabetic-width=22

------------------- Computation of the c-continuations (alphabet "ab01+.*")

Before the identification : 23 c-continuation(s)

l0=(a1+b2)*.(b3.a4.b5.a6.b7.(a8+b9)*.b10.a11.b12+b13.b14.a15.(a16+b17)*.b18.a19.b20).(a21+b22)*
l1=(a1+b2)*.(b3.a4.b5.a6.b7.(a8+b9)*.b10.a11.b12+b13.b14.a15.(a16+b17)*.b18.a19.b20).(a21+b22)*
l2=(a1+b2)*.(b3.a4.b5.a6.b7.(a8+b9)*.b10.a11.b12+b13.b14.a15.(a16+b17)*.b18.a19.b20).(a21+b22)*
l3=a4.b5.a6.b7.(a8+b9)*.b10.a11.b12.(a21+b22)*
l4=b5.a6.b7.(a8+b9)*.b10.a11.b12.(a21+b22)*
l5=a6.b7.(a8+b9)*.b10.a11.b12.(a21+b22)*
l6=b7.(a8+b9)*.b10.a11.b12.(a21+b22)*
l7=(a8+b9)*.b10.a11.b12.(a21+b22)*
l8=(a8+b9)*.b10.a11.b12.(a21+b22)*
l9=(a8+b9)*.b10.a11.b12.(a21+b22)*
l10=a11.b12.(a21+b22)*
l11=b12.(a21+b22)*
l12=(a21+b22)*
l13=b14.a15.(a16+b17)*.b18.a19.b20.(a21+b22)*
l14=a15.(a16+b17)*.b18.a19.b20.(a21+b22)*
l15=(a16+b17)*.b18.a19.b20.(a21+b22)*
l16=(a16+b17)*.b18.a19.b20.(a21+b22)*
l17=(a16+b17)*.b18.a19.b20.(a21+b22)*
l18=a19.b20.(a21+b22)*
l19=b20.(a21+b22)*
l20=(a21+b22)*
l21=(a21+b22)*
l22=(a21+b22)*

After the identification : 11 c-continuation(s)

L0={14} :    a.(a+b)*.b.a.b.(a+b)*
L1={0,1,2} :   (a+b)*.(b.a.b.a.b.(a+b)*.b.a.b+b.b.a.(a+b)*.b.a.b).(a+b)*
L2={7,8,9,15,16,17} : (a+b)*.b.a.b.(a+b)*
L3={12,20,21,22} :  (a+b)*
L4={5} :    a.b.(a+b)*.b.a.b.(a+b)*
L5={10,18} :   a.b.(a+b)*
L6={3} :    a.b.a.b.(a+b)*.b.a.b.(a+b)*
L7={6} :    b.(a+b)*.b.a.b.(a+b)*
L8={11,19} :   b.(a+b)*
L9={13} :    b.a.(a+b)*.b.a.b.(a+b)*
L10={4} :    b.a.b.(a+b)*.b.a.b.(a+b)*

------------------- The c-continuation quotient automaton
Initial state : 1
Final States : 3
Transitions :
State 0 :  a : 2
State 1 :  a : 1 b : 1 6 9
State 2 :  a : 2 b : 2 5
State 3 :  a : 3 b : 3
State 4 :  a : 7
State 5 :  a : 8
State 6 :  a : 10
State 7 :  b : 2
State 8 :  b : 3
State 9 :  b : 0
State 10 : b : 4
```

7.3. Comparison with Antimirov's construction

In Antimirov's construction, p partial derivatives are computed, among which n are distinct ones. While n is bounded by $||E||$, p is bounded by $||E||^2$ since each transition involves the computation of one partial derivative. Every partial derivative computation terminates with a syntactical equality check w.r.t. every distinct partial derivative already produced. The size of a partial derivative is $O(|E|^2)$. Hence an $O(||E||^3 \cdot |E|^2)$ worst-case time complexity for Antimirov's construction. Notice that the $O(||E||^2 \cdot |E|^2)$ complexity reported in [2] only covers the computation of the set of states.

In the c-continuation construction, this worst-case time complexity is reduced to $O(||E|| \cdot |E|^2)$ thanks to three facts: (1) the set of the $||E||$ c-continuations is computed at once, i.e. without syntactical equality check, (2) during the identification step, every c-continuation is compared to only one other one, thanks to a preliminary lexicographic sorting of the whole list of c-continuations, and (3) the set of transitions is obtained from the computation of the *First* sets of the subexpressions of E.

7.4. Improvements

The worst-case space and time complexity of the Algorithm *CtoE*3 can be reduced to $O(|E|^2)$ according to three main refinements:

(1) The star subexpressions of $\bar{E}$ are pre-processed: they are identified by names and their occurrences into c-continuations are replaced by these names. The resulting strings have a linear size w.r.t. $|E|$ and it is shown that they can be substituted to c-continuations inside the identification step.
(2) The First sets of the subexpressions of E are computed via a specific linking of the nodes of the syntax tree of E, with a time complexity linear w.r.t. $|E|$. This technique has already proved to be fruitful for devising an efficient algorithm to build the position automaton of an expression [15].
(3) The First set of a c-continuation is computed as a disjoint union of the First sets of some subexpressions of E.

These improvements lead to the $O(|E|^2)$ Algorithm *CtoE*2 which is analyzed in [6].

8. Conclusion

The c-continuation automaton of a regular expression is a "canonical" automaton, in the sense that the position automaton is a projection of this automaton and the equation automaton is a factorization.

The notion of c-continuation allows us to design an $O(|E|^2)$ space and time algorithm for the conversion of a regular expression into its equation automaton, improving by an $O(||E||^3)$ factor the partial derivative construction.

We are looking for reductions over the set of c-continuations, which would lead to smaller automata, with the same quadratic complexity.

References

[1] A.V. Aho, J.E. Hopcroft, J.D. Ullman, Data Structures and Algorithms, Addison-Wesley, Reading, MA, 1983.
[2] V. Antimirov, Partial derivatives of regular expressions and finite automaton constructions, Theoret. Comput. Sci. 155 (1996) 291–319.
[3] D. Beauquier, J. Berstel, P. Chrétienne, Éléments d'Algorithmique, Masson, Paris, 1992.
[4] G. Berry, R. Sethi, From regular expressions to deterministic automata, Theoret. Comput. Sci. 48 (1) (1986) 117–126.
[5] J.A. Brzozowski, Derivatives of regular expressions, J. Assoc. Comput. Mach. 11 (4) (1964) 481–494.
[6] J.-M. Champarnaud, D. Ziadi, From C-continuations to new quadratic algorithms for automaton synthesis, Internat. J. Algebra Comput. 11 (6) (2001) 707–735.
[7] J.-M. Champarnaud, D. Ziadi, From Mirkin's prebases to Antimirov's word partial derivatives, Fund. Inform. 45 (3) (2001) 195–205.
[8] V.M. Glushkov, The abstract theory of automata, Russian Math. Surveys 16 (1961) 1–53.
[9] J.E. Hopcroft, J.D. Ullman, Introduction to Automata Theory, Languages and Computation, Addison-Wesley, Reading, MA, 1979.
[10] S. Kleene, Representation of events in nerve nets and finite automata, Automata Studies Annals of Mathematics Studies Vol. 34, 1956, Princeton University Press, Princeton NJ, pp. 3–41.
[11] R.F. McNaughton, H. Yamada, Regular expressions and state graphs for automata, IEEE Trans. Electron. Comput. 9 (1960) 39–57.
[12] B.G. Mirkin, An algorithm for constructing a base in a language of regular expressions, Eng. Cybernet. 5 (1966) 110–116.
[13] R. Paige, R.E. Tarjan, Three partition refinement algorithms, SIAM J. Comput. 16 (6) (1987) 973–989.
[14] S. Yu, Regular languages, in: G. Rozenberg, A. Salomaa (Eds.), Handbook of Formal Languages, Vol. I, Words, Languages, Grammars, Springer, Berlin, 1997, pp. 41–110.
[15] D. Ziadi, J.-L. Ponty, J.-M. Champarnaud, Passage d'une expression rationnelle à un automate fini non-déterministe, Bull. Belg. Math. Soc. 4 (1997) 177–203.