From Mirkin's Prebases to Antimirov's Word Partial Derivatives

J.-M. Champarnaud and D. Ziadi
Université de Rouen, LIFAR
76821 Mont-Saint-Aignan Cedex, France
{champarnaud,ziadi}@dir.univ-rouen.fr

Abstract

Our aim is to give a proof of the fact that two notions related to regular expressions, the prebases due to Mirkin and the partial derivatives introduced by Antimirov lead to the construction of identical non-deterministic automata recognizing the language of a given regular expression.

Keywords: Regular expressions, Finite automata, Prebases, Partial derivatives.

1 Introduction

There are many sequential algorithms to convert a regular expression into an automaton (see for example [4]). We here consider two basic constructions, which yield non-deterministic automata: the first one lies on the notion of prebase of a regular expression due to Mirkin [9]; the second one is based on the notion of partial derivative of a regular expression introduced by Antimirov [1]. It turns out that the NFA's deduced from each of these notions are identical. We give a proof of this fact in this paper.

Mirkin and Antimirov constructions, as well as Brzozowski construction [3] are derived from the computation of a system of equations associated to the regular expression. The concept of word derivative developed by Brzozowski addresses deterministic systems while the notions of prebase and of partial derivative account for the non-deterministic case. The NFA yielded by Mirkin and Antimirov constructions will be called the equation automaton of the expression.

Our interest for the constructions which compute the equation automaton comes from its "small" size: its number of states is less than or equal to $n+1$, where n is the number of symbol occurrences in the expression, while the number of states of the classical position automaton (constructed by Glushkov [6] or McNaughton-Yamada [8] algorithms) is equal to $n+1$. In practice the gap between the two sizes can be arbitrarily large, as shown by the following example coming from the domain of programming languages compilation and cited in [1]. Let $L = \{A, \ldots, Z, a, \ldots, z\}$ be the set of letters, $D = \{0, \ldots, 9\}$ be the set of digits and $\Sigma = L \cup D$ be the alphabet. The set of identifiers of a programming language is typically represented by a regular expression over Σ such as $E = L \cdot (L + D)^*$. The associated lexical analyzer is either a 115-states position automaton or a 2-states equation automaton. Let us mention that the most efficient implementations to compute the equation automaton and the position automaton have the same worst case space and time complexity, which, as reported in [5], is quadratic on the size of the expression.

Section 1 recalls some useful notations, definitions and classical results of automata theory. Section 2 presents the general framework of regular expression equations. Mirkin's prebases are introduced in Section 3 and Antimirov's partial derivatives in Section 4. Section 5 provides a proof of the equivalence of the two approaches.

2 Preliminaries

Let us review basic notions and terminology concerning regular expressions, finite automata and derivatives. For further details, classical books [2, 7] or handbooks [11] are excellent references.

Let Σ be a non-empty finite set of *symbols*, called the *alphabet*. Symbols are denoted by $x_1, x_2, \ldots, x_m$. A *word* u over Σ is a finite sequence $(y_1, y_2, ..., y_n)$ of symbols, usually written $y_1 y_2 ... y_n$. The *length* of a word u, denoted $|u|$ is the number of symbols in u. The *empty word* denoted by ε has a null length. If $u = y_1 y_2 ... y_n$ and $v = z_1 z_2 ... z_p$ are two words over Σ, their *concatenation* $u \cdot v$, usually written uv, is the word $y_1 y_2 ... y_n z_1 z_2 ... z_p$. The set of all the *words* over Σ is denoted Σ^*. A *language* over Σ is a subset of Σ^*.

A *regular expression* over the alphabet Σ is 0, or 1, or a symbol $x_i \in \Sigma$, or is obtained by recursively applying the following rules: if F and G are two regular expressions, the *union* $(F + G)$, the *product* $(F \cdot G)$ (also written (FG)), and the *star* (F^*) are regular expressions. For example $(x_1 + x_1 \cdot x_2)^*$ is a regular expression over $\{x_1, x_2\}$. The *regular language* $L(E)$ denoted

by a regular expression E is such that: $L(0) = \emptyset$, $L(1) = \{\varepsilon\}$, $L(x_i) = \{x_i\}$ $\forall x_i \in \Sigma$, $L(F+G) = L(F) \cup L(G)$, $L(F \cdot G) = L(F)L(G)$ and $L(F^*) = L(F)^*$.

A regular expression over the alphabet Σ is a term of the algebra $\mathcal{T}_{Reg}$ 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 $\mathcal{T}_{Reg}$. Given a set S of identities, expression computations are performed on equivalence classes modulo S (see [1] for a detailed exposition). In the following, given a regular expression, we compare the computation of its Mirkin's prebase, and of its set of partial derivatives; we assume that an identical set of identities holds in the computation of both sets of expressions.

A *finite automaton* over Σ is a 5-tuple $\mathcal{M} = (Q, \Sigma, I, T, E)$ where Q is the set of *states*, I is the subset of *initial states*, T is the subset of *final states*, and E is the set of *transitions* which is a subset of $Q \times \Sigma \times Q$. $\mathcal{M}$ is *deterministic* ($\mathcal{M}$ is a DFA) if there is a unique initial state and if for all $(q, a) \in Q \times \Sigma$ there exists at most one state q' such that $(q, a, q') \in E$; otherwise $\mathcal{M}$ is a *NFA*. The transition set can be represented by a *transition function* δ from $Q \times \Sigma$ to Q if $\mathcal{M}$ is a DFA, and from $Q \times \Sigma$ to 2^Q if $\mathcal{M}$ is a NFA. A *path* of $\mathcal{M}$ is a sequence of consecutive transitions. A word $u = u_1 u_2 \dots u_n$ is *recognized* by $\mathcal{M}$ if it is the label of a path starting in I and ending in T. The language *recognized* by $\mathcal{M}$, denoted by $L_{\mathcal{M}}$, is the set of words $\mathcal{M}$ recognizes.

Finally, the following notations will be used. For a regular language L, $\lambda(L)$ is defined by: $\lambda(L) = \varepsilon$ if $\varepsilon \in L(E)$ and $\emptyset$ otherwise. Similarly, for a regular expression E, $\lambda(E)$ is defined by: $\lambda(E) = 1$ if $\varepsilon \in L(E)$ and 0 otherwise. We note $E \equiv F$ when the expressions E and F are syntactically identical. The size of E is denoted $|E|$. The alphabetic width of E, i.e. the number of symbol occurrences in E, is denoted $||E||$.

3 Systems of equations: from languages to expressions

Let L be a regular language and $\mathcal{M} = (Q, \Sigma, \{0\}, T, E)$ be a finite automaton such that $L_{\mathcal{M}} = L$. Assume that $\Sigma = \{x_1, x_2, \dots, x_m\}$ and $Q = [0, n]$. Let L_i, for all i in Q, be the language recognized by the automaton $(Q, \Sigma, \{i\}, T, E)$. Let us first recall how the L_i languages are related. The following *system of language equations* holds:

$$L = L_0$$

$$
\begin{aligned}
L_i &= (\bigcup_{j=1}^{m} x_j L_{ij}) \cup \lambda(L_i), \text{ for all } i \in [0,n] \\
L_{ij} &= \begin{cases} \bigcup_{k \in \delta(i,x_j)} L_k & \text{if } \mathcal{M} \text{ is a NFA} \\ L_k, \text{ with } k = \delta(i,x_j) & \text{if } \mathcal{M} \text{ is a DFA} \end{cases}
\end{aligned}
$$

Conversely, assume that the languages L_i, for $i \in [0,n]$, satisfy the system of equations S_L:

$$
\begin{aligned}
L &= L_0 \\
L_i &= (\bigcup_{j=1}^{m} x_j L_{ij}) \cup \lambda(L_i), \text{ for all } i \in [0,n] \\
L_{ij} &= \bigcup_{k \in I_{ij}} L_k, \text{ with } I_{ij} \subseteq [0,n]
\end{aligned}
$$

Let $\mathcal{L}$ be the automaton whose (1) states are languages L_i, (2) initial state is L, (3) final states are such that $\lambda(L_i) = \varepsilon$, (4) transitions are defined by: $L_k \in \delta(L_i, x_j) \Leftrightarrow k \in I_{ij}$. The automaton $\mathcal{L}$ recognizes the language L.

From a computational point of view, assume now that L is denoted by the regular expression E. The *system of expression equations* S_E deduces from S_L by replacing the language L_k (resp. L_{ij}) by a regular expression E_k (resp. E_{ij}) such that $L_k = L(E_k)$ (resp $L_{ij} = L(E_{ij})$).

$$
\begin{aligned}
E \equiv E_0 &= x_1 E_{01} + \cdots + x_m E_{0m} + \lambda(E_0) \\
E_1 &= x_1 E_{11} + \cdots + x_m E_{1m} + \lambda(E_1) \\
&\ldots \\
E_n &= x_1 E_{n1} + \cdots + x_m E_{nm} + \lambda(E_n) \\
E_{ij} &= \Sigma_{k \in I_{ij}} E_k, \text{ with } I_{ij} \subseteq [0,n]
\end{aligned}
$$

Let $\mathcal{E}$ be the automaton whose (1) states are expressions E_i, (2) initial state is E, (3) final states are such that $\lambda(E_i) = 1$, (4) transitions are defined by: $E_k \in \delta(E_i, x_j) \Leftrightarrow k \in I_{ij}$. The automaton $\mathcal{E}$ recognizes the language L.

Given a regular expression E, the practical problem is to find a finite (and as small as possible) set of expressions $\{E_1, \ldots, E_n\}$ such that S_E holds. In the deterministic case, the set of dissimilar word derivatives of E (Brzozowski [3]) is a solution. The notion of *base in the languages of regular*

expressions introduced by Spivak [10] leads to a similar solution. We now present two solutions for the non-deterministic case: Mirkin's prebases [9] and Antimirov's partial derivatives [1].

Example 1 *Let $E = x^*(xx+y)^*$. A non-deterministic equation system for E is the following:*

$$\begin{aligned} x^*(xx+y)^* &= x(x^*(xx+y)^*) + x(x(xx+y)^*) + y(xx+y)^* + 1 \\ x(xx+y)^* &= x(xx+y)^* \\ (xx+y)^* &= x(x(xx+y)^*) + y(xx+y)^* + 1 \end{aligned}$$

4 Mirkin's prebases

Mirkin's construction [9] is based on the notion of prebase of a regular expression. We first introduce the notion of *support*.

Definition 1 *Let E be a regular expression. The set $\pi \subseteq \{E_0, E_1, ..., E_n\}$, where $E_1, \ldots, E_n$ are non-null regular expressions, is called a support of E if the following equalities hold:*

$$\begin{aligned} E &\equiv E_0 \\ E_i &= x_1 E_{i1} + \cdots + x_m E_{im} + \lambda(E_i) \qquad \text{for } i = 0, \cdots, n \end{aligned}$$

where E_{ij}, for $i = 0, \cdots, n$ and $j = 1, \cdots, m$, is a linear combination of elements of the set π.

Definition 2 *If the set π is a support of E, then the set $\Pi = \pi \cup \{E\}$ is a prebase of E.*

Notice that for a non null regular expression E, any prebase of E contains E (for the null expression, there exists a unique and empty support). On the other hand, a support of E contains E only if E appears in the linear combination associated to some expression E_{ij}. Therefore the property: $|\Pi| - |\pi| \leq 1$ holds for any expression E.

Let $E_{ij} = \Sigma_{k \in I_{ij}} E_k$, with $I_{ij} \subseteq [0, n]$. We'll use the following notation: X is the mapping defined by $X(E_{ij}) = \{E_k \mid k \in I_{ij}\}$.

Example 2 *Consider the system of equations given in Example 1 for the expression* $E = x^*(xx + y)^*$. *The set* $\{x^*(xx + y)^*,\ x(xx + y)^*,\ (xx + y)^*\}$ *is both a support and a prebase of* E.

Let R be a set of non-null regular expressions and F be a regular expression. We define the operation $\odot$ as follows: **if** $F \neq 0$ **then** $F \odot R = \{F \cdot G \mid G \in R\}$ and $R \odot F = \{G \cdot F \mid G \in R\}$ **else** $F \odot R = R \odot F = \emptyset$. We'll use this operation to formulate the next three propositions which are a rewriting of Mirkin's paper.

Proposition 1 (Mirkin [9]) *Let* E *be a regular expression. Then the set* π_E, *recursively computed as follows, is a support of* E:

$$\begin{array}{ll} [E = 0 \text{ or } E = 1] & \pi_E = \emptyset \\ [E = x_i] & \pi_E = \{1\} \quad \forall x_i \in \Sigma \\ [E = F + G] & \pi_E = \pi_F \cup \pi_G \\ [E = F \cdot G] & \pi_E = \pi_F \odot G \cup \lambda(F) \odot \pi_G \\ [E = F^*] & \pi_E = \pi_F \odot F^* \end{array}$$

Proof. For $E = 0$ and $E = 1$ we have the following equation:

$$E = x_1 \cdot 0 + \cdots + x_m \cdot 0 + \lambda(E)$$

Thus the empty set is a support of $E = 0$ and of $E = 1$.

For $E = x_k$, with $x_k \in \Sigma$, we have the following system:

$$\left\{ \begin{array}{rcl} x_k & = & x_1 \cdot 0 + \cdots + x_k \cdot 1 + \cdots + x_m \cdot 0 \\ 1 & = & x_1 \cdot 0 + \cdots + x_m \cdot 0 + 1 \end{array} \right.$$

Thus the set $\{1\}$ is a support of $E = x_k$, for all x_k in Σ.

Let $\pi_F = \{F_1, \cdots, F_n\}$ be the support associated to the regular expression F, and $\pi_G = \{G_1, \cdots, G_\ell\}$ be the support associated to G. By induction hypothesis, we have:

$$\begin{array}{rcll} F & \equiv & F_0 & \\ F_i & = & x_1 F_{i1} + \cdots + x_m F_{im} + \lambda(F_i) & \text{for } i = 0, \cdots, n \\ F_{ij} & = & \Sigma_{k \in I_{ij}} F_k, \text{ with } I_{ij} \subseteq [0, n] & \\ \textit{and} & & & \end{array}$$

$$\begin{array}{rcl} G & \equiv & G_0 \\ G_k & = & x_1 G_{k1} + \cdots + x_m G_{km} + \lambda(G_k) \quad \text{for } k = 0, \cdots, \ell \\ G_{ij} & = & \Sigma_{k \in J_{ij}} G_k, \text{ with } J_{ij} \subseteq [0, \ell] \end{array}$$

Case 1: $E = F + G$. Since $F \equiv F_0$ and $G \equiv G_0$, we have $E \equiv E_0 \equiv F_0 + G_0$ and the following equation holds:

$$E_0 = x_1(F_{01} + G_{01}) + \cdots + x_m(F_{0m} + G_{0m}) + \lambda(F_0 + G_0)$$

Since for $j = 1, \cdots, m$, F_{0j} (resp. G_{0j}) is a linear combination of expressions of π_F (resp. of π_G), $F_{0j} + G_{0j}$ is a linear combination of expressions of $\pi_F \cup \pi_G$. Thus the set $\pi_F \cup \pi_G$ is a support of $E = F + G$.

Case 2: $E = F \cdot G$. We have $E \equiv E_0 \equiv F_0 \cdot G_0$. The following equation holds:

$$E_0 \quad = \quad x_1 F_{01} G_0 + \cdots + x_m F_{0m} G_0 + \lambda(F_0) G_0$$

Since $F_{01}, \ldots, F_{0m}$ are linear combinations of expressions of π_F, $F_{01}G_0, \ldots, F_{0m}G_0$ are linear combinations of expressions of $\pi_F \odot G_0$. Thus the set $\pi_F \odot G_0 \cup \lambda(F_0) \odot \pi_G$ is a support of $E = F \cdot G$.

Case 3: $E = F^*$. We have:

$$\begin{array}{rcl} E_0 & = & F_0^* \\ & = & (F_0 \setminus 1) \cdot F_0^* + \lambda(F_0^*) \\ & = & (x_1 F_{01} + \cdots + x_m F_{0m}) \cdot F_0^* + \lambda(F_0^*) \\ & = & x_1 F_{01} F_0^* + \cdots + x_m F_{0m} F_0^* + \lambda(F_0^*) \end{array}$$

Thus the set $\pi_F \odot F^*$ is a support of $E = F^*$. ■

Example 3 *Let $E = x^*(xx + y)^*$. The following supports and prebases are successively computed:*

$$\begin{array}{rcl} \pi_x & = & \{1\} \\ \pi_{x^*} & = & \{x^*\} \\ \pi_{xx} & = & \{x\} \end{array}$$

$$\begin{aligned}
\pi_{xx+y} &= \{x, 1\} \\
\pi_{(xx+y)^*} &= \{x(xx+y)^*, (xx+y)^*\} \\
\pi_{x^*(xx+y)^*} &= \{x^*(xx+y)^*, x(xx+y)^*, (xx+y)^*\} \\
\Pi_{x^*(xx+y)^*} &= \{x^*(xx+y)^*, x(xx+y)^*, (xx+y)^*\}
\end{aligned}$$

Proposition 2 (Mirkin [9]) *Let E be a regular expression. The following property holds:*

$$|\Pi_E| \leq ||E|| + 1$$

Proof. We prove that: $|\pi_E| \leq ||E||$, which is equivalent, since $|\Pi_E| - |\pi_E| \leq 1$. The proof is by induction on the length of E. It is easy to see that for $|E| \leq 1$ the proposition is true. Indeed $|\pi_0| = 0 = ||0||$, $|\pi_1| = 0 = ||1||$ and $|\pi_{x_i}| = 1 = ||x_i||$, for all $x_i \in \Sigma$.

Assume that the proposition is true for any expression of length less or equal to n, $n \geq 2$. Let F and G two regular expressions such that: $|F| \leq n$ and $|G| \leq n$. By induction hypothesis, we have $|\pi_F| \leq ||F||$ and $|\pi_G| \leq ||G||$. We consider the following three cases:

Case 1: $E = F+G$. In this case $\pi_E = \pi_F \cup \pi_G$. Hence $|\pi_E| \leq |\pi_F| + |\pi_G| \leq ||F|| + ||G|| = ||E||$.

Case 2: $E = F \cdot G$. We have $\pi_E = \pi_F \odot G \cup \lambda(F) \odot \pi_G$. Hence $|\pi_E| \leq (|\pi_F \odot G| + |\lambda(F) \odot \pi_G|) \leq |\pi_F| + |\pi_G| \leq ||F|| + ||G|| = ||E||$.

Case 3: $E = F^*$. We have $\pi_E = \pi_F \odot E$. Thus $|\pi_E| = |\pi_F| \leq ||F|| = ||E||$.

■

Proposition 3 (Mirkin [9]) *Let E be a regular expression and Π_E be a base of E. The language $L(E)$ is recognized by the automaton $\mathcal{M}_E = (Q, \Sigma, q_0, T, \delta)$ such that:*

$$\begin{aligned}
Q &= \Pi_E = \{E, E_1, \ldots, E_n\} \\
q_0 &= E \\
T &= \{E_i \in Q \mid \lambda(E_i) = 1\} \\
\delta(E_i, x_j) &= X(E_{ij}) \quad \textit{for all } 0 \leq i \leq n \ , \ 1 \leq j \leq m
\end{aligned}$$

5 Antimirov's partial derivatives

Let us consider the equations:

$$\begin{aligned}
E \equiv E_0 &= x_1 E_{01} + \cdots + x_m E_{0m} + \lambda(E_0) \\
E_{0j} &= \Sigma_{k \in I_{0_j}} E_k, \text{ with } I_{0j} \subseteq [0, n]
\end{aligned}$$

In the deterministic solution due to Brzozowski, E_{0j} is set to $x_j^{-1}E$ and $x_j\Sigma_{k\in I_{0j}}E_k$ is seen as a unique term. In Antimirov's solution, $x_j\Sigma_{k\in I_{0j}}E_k$ is seen as the sum of the expressions x_jE_k, with $k \in I_{0j}$. The expressions E_k, for $k \in I_{0j}$, are called the *partial derivatives* of E w.r.t. the symbol x_j; their sum $\Sigma_{k\in I_{0j}}E_k$ behaves in the same manner as does the $x_j^{-1}E$ derivative in the deterministic version.

The set $\{E_k \mid k \in I_{0j}\}$ of the partial derivatives of E w.r.t. x_j is denoted $\partial_{x_j}(E)$. Antimirov provides both a formal definition of the notion of partial derivative and a set of rules to compute the set of partial derivatives of a regular expression.

Definition 3 (set of partial derivatives w.r.t. a symbol) *Given a regular expression E and a symbol x_j, the set $\partial_{x_j}(E)$ of the partial derivatives of E w.r.t. x_j is recursively defined on the structure of E as follows:*

$$
\begin{aligned}
\partial_{x_j}(0) &= \emptyset \\
\partial_{x_j}(1) &= \emptyset \\
\partial_{x_j}(x) &= \begin{cases} \{1\} & \text{if } x_j = x \\ \emptyset & \text{otherwise} \end{cases} \\
\partial_{x_j}(F+G) &= \partial_{x_j}(F) \cup \partial_{x_j}(G) \\
\partial_{x_j}(F\cdot G) &= \begin{cases} \partial_{x_j}(F) \odot G & \text{if } \lambda(F) = 0 \\ \partial_{x_j}(F) \odot G \cup \partial_{x_j}(G) & \text{otherwise} \end{cases} \\
\partial_{x_j}(F^*) &= \partial_{x_j}(F) \odot F^*
\end{aligned}
$$

Let us point out that we have slightly modified the original definition: the $\odot$ operator is introduced in $\partial_{x_j}(F\cdot G)$ and $\partial_{x_j}(F^*)$ formulas instead of the $\cdot$ operator, since sets of partial derivatives should not contain the null expression.

The symbol x_j in $\partial_{x_j}(E)$ can be replaced by any word u of Σ^* or by any set of words U, according to the formulas:

$$
\begin{aligned}
\partial_{\varepsilon}(E) &= \{E\} \\
\partial_{ux_j}(E) &= \partial_{x_j}(\partial_u(E)) \\
\partial_U(E) &= \bigcup_{u\in U} \partial_u(E)
\end{aligned}
$$

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

Example 4 *Let $E = x^*(xx+y)^*$. The computation of the partial derivatives is as follows:*

$$\begin{array}{rcl}
\partial_\varepsilon(E) = E & = & x^*(xx+y)^* \\
\partial_x(E) & = & \partial_x(x^*)(xx+y)^* \cup \partial_x((xx+y)^*) \\
& = & \partial_x(x)x^*(xx+y)^* \cup (\partial_x(xx) \cup \partial_x(y))(xx+y)^* \quad = \\
& = & \{x^*(xx+y)^*\} \cup \{x(xx+y)^*\} \\
& = & \{x^*(xx+y)^*,\ x(xx+y)^*\} \\
\partial_y(E) & = & \partial_y(x^*)(xx+y)^* \cup \partial_y((xx+y)^*) \\
& = & \partial_y(x)x^*(xx+y)^* \cup (\partial_y(xx) \cup \partial_y(y))(xx+y)^* \\
& = & \emptyset \cup (\emptyset \cup \{1\})(xx+y)^* \\
& = & \{(xx+y)^*\} \\
\partial_x(x(xx+y)^*) & = & \partial_x(x)(xx+y)^* \quad = \quad \{(xx+y)^*\} \\
\partial_y(x(xx+y)^*) & = & \partial_y(x)(xx+y)^* \quad = \quad \emptyset \\
\partial_x((xx+y)^*) & = & \partial_x(xx+y)(xx+y)^* \quad = \quad (\partial_x(xx) \cup \partial_x(y))(xx+y)^* \\
& = & \{x(xx+y)^*\} \\
\partial_y((xx+y)^*) & = & \partial_y(xx+y)(xx+y)^* \quad = \quad (\partial_y(xx) \cup \partial_y(y))(xx+y)^* \\
& = & \{(xx+y)^*\}
\end{array}$$

Hence $\mathcal{PD}(E) = \{x^*(xx+y)^*,\ x(xx+y)^*,\ (xx+y)^*\}$.

Proposition 4 (Antimirov [1]) *The cardinality of the set $\mathcal{PD}(E)$ of all partial derivatives of a regular expression E is less than or equal to $\|E\|+1$.*

Proposition 5 (Antimirov [1]) *Let E be a regular expression. The language $L(E)$ is recognized by the automaton $\mathcal{A}_E = (Q, \Sigma, q_0, T, \delta)$ such that:*

$$\begin{array}{rcl}
Q & = & \mathcal{PD}(E) \\
q_0 & = & E \\
T & = & \{q \in Q \mid \lambda(q) = 1\} \\
\delta(p, x_j) & = & \partial_{x_j}(p) \quad \textit{, for all } p \in Q \textit{ and } 1 \leq j \leq m
\end{array}$$

6 Prebases vs partial derivatives

Putting together Proposition 1 and Definition 3 shows that Mirkin's prebases and Antimirov's partial derivatives lead to identical systems of expression equations and thus to identical non-deterministic automata. Example 3,

based on a bottom-up computation, and Example 4, based on a top-down one, illustrate this fact. The following theorem gives a formal proof of the equivalence of the two constructions.

Theorem 1 *Let E be a regular expression. Consider the support $\pi_E \subseteq \{E_0, E_1, ..., E_n\}$, which is such that:*

$$\begin{aligned} E &= x_1 E_{01} + x_2 E_{02} + \cdots + x_m E_{0m} + \lambda(E) \\ E_{0j} &= \Sigma_{k \in I_{0j}} E_k, \text{ with } I_{0j} \subseteq [0, n] \text{ for all } 1 \leq j \leq m \end{aligned}$$

Then the following property holds:

$$\partial_{x_j}(E) = X(E_{0j}), \text{ for all } x_j \text{ in } \Sigma$$

Proof. The proof is by induction on the length of E. For the base cases ($|E| \leq 1$) the situation is the following:

$$\begin{array}{lll} [E = 0] & E = x_1 \cdot 0 + \cdots + x_j \cdot 0 + \cdots + x_m \cdot 0 + 0 & \partial_{x_j}(E) = \emptyset = X(E_{0j}) \\ [E = 1] & E = x_1 \cdot 0 + \cdots + x_j \cdot 0 + \cdots + x_m \cdot 0 + 1 & \partial_{x_j}(E) = \emptyset = X(E_{0j}) \\ [E = x_j] & E = x_1 \cdot 0 + \cdots + x_j \cdot 1 + \cdots + x_m \cdot 0 + 0 & \partial_{x_j}(E) = \{1\} = X(E_{0j}) \\ [E = x_i] & E = x_1 \cdot 0 + \cdots + x_j \cdot 0 + \cdots + x_m \cdot 0 + 0 & \partial_{x_j}(E) = \emptyset = X(E_{0j}) \end{array}$$

Assume that the proposition is true for any expression of length less or equal to n, $n \geq 2$. Let E be a regular expression of length $n+1$. We consider three cases.

Case 1: $E = F + G$. Let $\pi_F = \{F_1, \cdots, F_n\}$ be the support of F and $\pi_F = \{F_1, \cdots, F_n\}$ be the support of G. We have:

$$\begin{aligned} F &= x_1 F_{01} + \cdots + x_j F_{0j} + \cdots + x_m F_{0m} + \lambda(F) \\ G &= x_1 G_{01} + \cdots + x_j G_{0j} + \cdots + x_m G_{0m} + \lambda(G) \end{aligned}$$

By induction hypothesis it comes:

$$\partial_{x_j}(F) = X(F_{0j}) \text{ and } \partial_{x_j}(G) = X(G_{0j}), \text{ for all } 1 \leq j \leq m$$

Let us consider the expression $E = F + G$.

From $E = x_1(F_{01}+G_{01}) + \cdots + x_j(F_{0j}+G_{0j}) + \cdots + x_m(F_{0m}+G_{0m}) + \lambda(E)$, we deduce:

$$\begin{aligned} X(F_{0j} + G_{0j}) &= X(F_{0j}) \cup X(G_{0j}) \\ &= \partial_{x_j}(F) \cup \partial_{x_j}(G) \\ &= \partial_{x_j}(F + G) = \partial_{x_j}(E) \end{aligned}$$

Case 2: $E = F \cdot G$. In this case we have:

$$\begin{aligned} E &= x_1 F_{01} G + \cdots + x_j F_{0j} G + \cdots + x_m F_{0m} G + \lambda(F) G \\ &= x_1 F_{01} G + \cdots + x_j F_{0j} G + \cdots + x_m F_{0m} G + \\ &\quad \lambda(F)(x_1 G_{01} + \cdots + x_j G_{0j} + \cdots + x_m G_{0m} + \lambda(G)) \\ &= x_1(F_{01} G + \lambda(F) G_{01}) + \cdots + x_j(F_{0j} G + \lambda(F) G_{0j}) + \cdots \\ &\quad + x_m(F_{0m} G + \lambda(F) G_{0m}) + \lambda(F)\lambda(G) \end{aligned}$$

Thus we have:

$$\begin{aligned} X((F_{0j} G + \lambda(F) G_{0j})) &= X(F_{0j} G) \cup X(\lambda(F) G_{0j}) \\ &= X(F_{0j}) \odot G \cup \lambda(F) \odot X(G_{0j}) \\ &= \partial_{x_j}(F) \odot G \cup \lambda(F) \odot \partial_{x_j}(G) \end{aligned}$$

Case 3: $E = F^*$. We have:

$$\begin{aligned} E &= F^* \\ &= (F \setminus 1) \cdot F^* + \lambda(F^*) \\ &= (x_1 F_{01} + \cdots + x_j F_{0j} + \cdots + x_m F_{0m}) \cdot F^* + \lambda(F^*) \\ &= x_1 F_{01} F^* + \cdots + x_j F_{0j} F^* + \cdots + x_m F_{0m} F^* + \lambda(F^*) \end{aligned}$$

We deduce that:

$$\begin{aligned} X(F_{0j} F) &= X(F_{0j}) \odot F^* \\ &= \partial_{x_j}(F) \odot F^* \end{aligned}$$

■

7 Conclusion

The computation of the Mirkin's prebases and the Antimirov's partial derivatives of a regular expression lead to the same "coefficients" E_{ij} in the formal system of equations associated to the expression, and therefore to the same NFA recognizer. Let us point out that the notion of partial derivative provides a solid theoretical framework for the study of finite automaton constructions.

References

[1] V. Antimirov. Partial derivatives of regular expressions and finite automaton constructions. *Theoret. Comput. Sci.*, 155:291–319, 1996.

[2] D. Beauquier, J. Berstel, and P. Chrétienne. *Éléments d'Algorithmique*. Masson, Paris, 1992.

[3] J. A. Brzozowski. Derivatives of regular expressions. *J. Assoc. Comput. Mach.*, 11(4):481–494, 1964.

[4] J. -M. Champarnaud , J.-L. Ponty and D. Ziadi. From regular expressions to finite automata, *Intern. J. Computer Math.*, vol. **72**, 415–431, 1999.

[5] J. -M. Champarnaud and D. Ziadi. From C-Continuations to New Quadratic Algorithms for Automaton Synthesis. *Intern. Journ. of Alg. Comp.*, to appear.

[6] V. M. Glushkov. The abstract theory of automata. *Russian Mathematical Surveys*, 16:1–53, 1961.

[7] J. E. Hopcroft and J. D. Ullman. *Introduction to Automata Theory, Languages and Computation*. Addison-Wesley, Reading, MA, 1979.

[8] R. F. McNaughton and H. Yamada. Regular expressions and state graphs for automata. *IEEE Transactions on Electronic Computers*, 9:39–57, March 1960.

[9] B. G. Mirkin. An algorithm for constructing a base in a language of regular expressions. *Engineering Cybernetics*, 5:110–116, 1966.

[10] M. A. Spivak, An Algorithm for the Abstract Synthesis of Automata for an Extended Language of Regular Expressions, (in russian), *Izv. Akad. Nauk SSSR, Techn. Kibernet.* **1** (1965) 51–57. English translation in *Engineering Cybernetics*.

[11] S. Yu. Regular languages. In G. Rozenberg and A. Salomaa, editors, *Handbook of Formal Languages*, volume I, Words, Languages, Grammars, pages 41–110. Springer-Verlag, Berlin, 1997.