HOW EXPRESSIONS CAN CODE FOR AUTOMATA

Sylvain Lombardy[1] and Jacques Sakarovitch[2]

Abstract. In this paper we investigate how it is possible to recover an automaton from a rational expression that has been computed from that automaton. The notion of derived term of an expression, introduced by Antimirov, appears to be instrumental in this problem. The second important ingredient is the co-minimization of an automaton, a dual and generalized Moore algorithm on non-deterministic automata.

We show here that if an automaton is then sufficiently "decorated", the combination of these two algorithms gives the desired result. Reducing the amount of "decoration" is still the object of ongoing investigation.

1991 Mathematics Subject Classification. 68Q45, 68Q70.

dedicated to Imre Simon on the occasion of his 60th birtday

1. A natural question

Kleene's theorem states the *equality of two families of languages*: the family of langagues described by rational (*i.e.* regular) expressions coincides with the family of languages accepted (or recognized) by finite automata — equality which is often written as:

$$\mathrm{Reg}\,A^* = \mathrm{Rec}\,A^* \,.$$

The proof of this equality amounts to showing the two inclusions:

$$\mathrm{Rec}\,A^* \subseteq \mathrm{Reg}\,A^* \qquad (1a) \quad \text{and} \qquad \mathrm{Reg}\,A^* \subseteq \mathrm{Rec}\,A^* \qquad (1b)$$

and is *constructive* — as usual in the field. In the earlier proofs, inclusion (1a) is established by an algorithm, say Φ, that takes an automaton $\mathcal{A}$ and produces a

Keywords and phrases: finite automata, regular expression, derivation of expressions, quotient of automata.

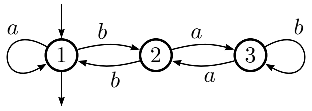
[1] LIAFA (UMR 7089), Université Paris 7, 2 place Jussieu, 75251 Paris cedex 05, France
e-mail: `lombardy@liafa.jussieu.fr`

[2] LTCI (UMR 5141), CNRS / ENST, 46 rue Barrault, 75634 Paris Cedex 13, France
e-mail: `sakarovitch@enst.fr`

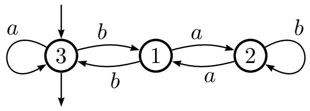
rational expression E — we thus can write $\mathsf{E} = \Phi(\mathcal{A})$ — such that the language denoted by E is equal to the language accepted by $\mathcal{A}$. And conversely, inclusion (1b) is obtained by showing that $\operatorname{Rec} A^*$ is (effectively) closed under union, product and star.

This closure proof is easily turned into an algorithm, say Ψ, that takes an expression E and computes an automaton $\mathcal{A} = \Psi(\mathsf{E})$ with the property that the language accepted by $\mathcal{A}$ is equal to the language denoted by E. It was not long before it was understood that these algorithms and their properties are as interesting in themselves as to be a piece of the proof of Kleene's theorem.

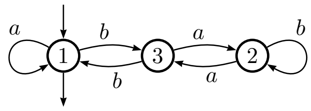
The problem we address here is to find Φ-type and Ψ-type algorithms which would be *inverse* of each other, that is which are going forth and back between expressions and automata not only at the level of the families but at the level of the individual objects. In order to understand the challenge of this problem, we have to say more about the Φ-type and Ψ-type algorithms.



(a) $\mathsf{E}_1 = a^*{+}a^*b(ba^*b)^*ba^*{+}a^*b(ba^*b)^*a(b{+}a(ba^*b)^*a)^*a(ba^*b)^*ba^*$



(b) $\mathsf{E}_2 = (a + bb + ba(b + aa)^*ab)^*$



(c) $\mathsf{E}_3 = a^* + a^*b(ab^*a + ba^*b)^*ba^*$

FIGURE 1. The state elimination method on $\mathcal{P}_1$, the " divisor by 3"

The two better known algorithms of the Φ-type (*i.e.* from automata to expressions) are the so-called "McNaughton-Yamada" algorithm (*cf.* [13]) and "state elimination method" (*cf.* [17, 18] for instance)[1]. Although the computations involved in these algorithms are somewhat different (above all they are organized in a different way), they produce roughly the same results. Both algorithms depend on *an ordering* of the states of the automaton. In this paper we take for Φ the state elimination method. Figure 1 shows the results of the state elimination method on a same automaton, with three different orderings of the states. The result, and in particular its size, may considerably vary with the ordering that is used. But one cannot avoid a combinatorial explosion in the general case[2]:

Fact 1.1. *The size of a rational expression* E *computed from a finite automaton* $\mathcal{A}$ *by the state elimination method may be exponential in the number of states of* $\mathcal{A}$.

[1]A third one (*cf.* [10]) gives rise to elegant proofs but is not useful for actual computations.

[2]*e.g.* an automaton whose underlying graph is the complete graph on the set of states.

There is a larger variety of algorithms turning a rational expression into a finite automaton — that is Ψ-type algorithms — both in results and in methods, than those of Φ-type. They fall roughly into two families.

The first class of algorithms yields what is often called the *Glushkov*, or the *position*, *automaton* of an expression [11]. It is a non deterministic automaton with $n+1$ states for an expression of *literal length* n. The Thompson construction [16] produces an automaton with ε-moves which is transformed into the position automaton when the ε-moves are eliminated in the adequate way. Let us denote by Ψ_{p} an algorithm that produces the position automaton.

The algorithms of the second class are based on the definition of the *derivation of an expression*. First introduced by Brzozowski [6], the definition of derivation has been slightly, but smartly, modified by Antimirov [1] and yields a non deterministic automaton which we propose to call the *derived term automaton* of the expression and which is smaller than or equal to the position automaton. The automaton of derived expressions computed in [6] is the determinized automaton of the derived term automaton. Champarnaud and Ziadi [8] have given an efficient method to compute the derived term automaton of an expression.

A bridge between the two families of algorithms was first given by Berry–Sethi who showed that the Brzozowski derivation applied on a “ linearized” version of an expression gives the position automaton of that expression [3, 4], and then by Champarnaud–Ziadi who showed that the derived term automaton of an expression is a *quotient* (*i.e.* a morphic image) of the position automaton [9].

Fact 1.2. *In the worst case, the minimal size (number of states) of an automaton accepting the language denoted by an expression is linear in the literal length of the expression*[3].

The juxtaposition of Facts 1.1 and 1.2 shows that there is no hope to find algorithms which are inverse of each other if we stay in these general families.

In [7], Caron and Ziadi have tackled a problem which proves to be closely related to ours. More precisely, they describe an algorithm, say Θ, which decides whether or not an automaton $\mathcal{A}$ is *the* position automaton of a rational expression E; and if the answer is positive, Θ moreover computes an expression which is *almost* E, namely the *star normal form* of E as defined by Brüggemann-Klein [5]. Even if Θ is not properly a Φ-type algorithm since it does not compute an expression for every automaton, it holds:

$$\text{For any star normal form rational expression } \mathsf{E}, \quad \Theta(\Psi_{\mathsf{p}}(\mathsf{E})) = \mathsf{E}\,. \tag{1}$$

Our purpose here is to describe a (slight) modification of Φ into Φ' and an algorithm Ω which given a rational expression E computes an equivalent automaton and such that, if E is obtained from an automaton $\mathcal{A}$ by a Φ'-type algorithm, then the result of Ω is precisely $\mathcal{A}$:

$$\text{For any automaton } \mathcal{A}, \quad \Omega(\Phi'(\mathcal{A})) = \mathcal{A}\,. \tag{2}$$

[3]An example of such a worst case is given by $\mathsf{E} = f$ where f is a word.

The paper is organized as follow. In the next section, we present the two main constructions on which such an Ω is built: the — barely modified — *(Antimirov) derivation of expressions* and the *co-minimization* of an automaton. In Section 3 we show how these two constructions can be combined in order to give the core of the algorithm Ω and how Φ has to be transformed into Φ' by a kind of coding in order that (2) holds (Theorem 3.6). It is noteworthy that the main property on which Theorem 3.6 relies gives an algorithm that computes a *deterministic* automaton from an expression itself obtained from a deterministic automaton *without* any determinization algorithm (Corollary 3.7). The last section presents three directions of research for reducing the "gap" between Φ and Φ'. They are based on the observation of characteristic examples and by this the paper shows that the "natural" question we have raised is not yet completely answered but still the object of a work in progress.

2. The ingredients for a solution

As we just said, the algorithm Ω is based on the Antimirov derivation of expressions and on the co-minimization of automata. We have first to recall what are these two operations.

In the sequel, A is an alphabet, *i.e.* a finite set of letters and A^* the free monoid generated by A. Rational expressions over A^* are the well-formed formulae with $\mathsf{0}$, $\mathsf{1}$, $a \in A$ as atomic formulae, $*$ as unary operator and $+$ and $\cdot$ as binary operators. The *constant term* of an expression E, denoted by $\mathsf{c}(\mathsf{E})$, is (the Boolean) $\mathsf{1}$ or $\mathsf{0}$ according to whether the empty word belongs or not to the language denoted by E. It can easily be computed directly on the rational expression, without computing the language denoted by the expression (*cf.* [1,12] for instance). The *literal length* of E is the number of occurrences of letters of A in E.

2.1. The automaton of derived terms

An algebraic characterization of rational languages is that every rational language has a finite number of left quotients. The purpose of "Brozozowski" derivatives is to lift that characterization at the level of expressions [6]. Antimirov partial derivatives achieve the same lifting in an indirect but more efficient way. To an expression E that denotes a language L is associated a finite set $\mathcal{T}$ of expressions — which we call *derived terms* of E — such that any left quotient of L is a *union*[4] of some of the languages denoted by the expressions in $\mathcal{T}$ [1]. The automaton build with the "Brozozowski" derivatives is the determinized of the automaton build with the derived terms.

[4]To tell the truth, the notion of derived terms is better understood when expressed in the larger framework of power series — languages being series with coefficients in the Boolean semiring — and of expressions *with multiplicity* (*cf.* [12]).

A series s is rational — *i.e.* denoted by a rational expression E — iff it is contained in a *finitely generated* module (of series) U which is *closed under left quotient*. The derived terms of E are expressions that denote a set of generators of U.

The following definitions give a procedure for computing the derived terms of an expression.

Definition 2.1 (**Antimirov** [1])**.** Let E be a rational expression on A and let a be a letter in A. The $\mathbb{B}$-*derivative*[5] of E with respect to a, denoted $\frac{\partial}{\partial a}\mathsf{E}$, is *a set of rational expressions on* A, recursively defined by:

$$\frac{\partial}{\partial a}\mathsf{0} = \frac{\partial}{\partial a}\mathsf{1} = \emptyset\,,$$

$$\forall a,b \in A \quad \frac{\partial}{\partial a}b = \begin{cases} \{1\} & \text{if} \quad b=a \\ \emptyset & \text{otherwise} \end{cases}$$

$$\frac{\partial}{\partial a}(\mathsf{E}+\mathsf{F}) = \frac{\partial}{\partial a}\mathsf{E} \cup \frac{\partial}{\partial a}\mathsf{F} \tag{3}$$

$$\frac{\partial}{\partial a}(\mathsf{E}\cdot\mathsf{F}) = \left[\frac{\partial}{\partial a}\mathsf{E}\right]\cdot\mathsf{F} \cup \mathsf{c}(\mathsf{E})\,\frac{\partial}{\partial a}\mathsf{F} \tag{4}$$

$$\frac{\partial}{\partial a}(\mathsf{E}^*) = \left[\frac{\partial}{\partial a}\mathsf{E}\right]\cdot\mathsf{E}^* \tag{5}$$

The induction implied by $(3-5)$ should be interpreted by distributing derivation and product over union:

$$\frac{\partial}{\partial a}\left[\bigcup_{i\in I}\mathsf{E}_i\right] = \bigcup_{i\in I}\frac{\partial}{\partial a}\mathsf{E}_i\,, \qquad \left[\bigcup_{i\in I}\mathsf{E}_i\right]\cdot\mathsf{F} = \bigcup_{i\in I}(\mathsf{E}_i\cdot\mathsf{F})\,.$$

Definition 2.2. Let E be a rational expression on A and g a non empty word of A^*, *i.e.* $g = f\,a$ with a in A. The $\mathbb{B}$-*derivative* of E with respect to g, denoted $\frac{\partial}{\partial g}\mathsf{E}$, is *the set of rational expressions over* A, recursively defined by formulae (3) – (5) and by:

$$\forall f \in A^+\,,\ \forall a \in A \quad \frac{\partial}{\partial f a}\mathsf{E} = \frac{\partial}{\partial a}\left(\frac{\partial}{\partial f}\mathsf{E}\right)\,. \tag{6}$$

We shall call *derived term* of E the expression E itself or any rational expression which belongs to a set $\frac{\partial}{\partial g}\mathsf{E}$ for some g in A^+.

Remark 2.3. Contrary to the derivation defined by Brzozowski [6], the result of the $\mathbb{B}$-derivation of a rational expression is not an expression but a *set* of rational expressions.

Theorem 2.4 (Antimirov [1])**.** *The number of derived terms of a rational expression* E *is finite and smaller than or equal to the literal length of* E *plus 1.*

---

[5]We call it "$\mathbb{B}$-derivative" and not simply "derivative" for two reasons. First in order to avoid confusion with the derivation defined by Brzozowski, and second because the formulae depend on the semiring of multiplicities and can be defined for other semirings (*cf.* [12]).

Example 2.5. Let $\mathsf{E}_2 = \big(a + bb + ba(b+aa)^*ab\big)^*$ be the expression computed in Figure 1 (b). The computation of the derived terms of E_2 goes as follow [for conciseness we put $\mathsf{H}_2 = (b+aa)^*ab$]:

$$\frac{\partial}{\partial a}\,\mathsf{E}_2 = \{\mathsf{E}_2\}\,, \quad \frac{\partial}{\partial b}\,\mathsf{E}_2 = \{b\,\mathsf{E}_2, a\,\mathsf{H}_2\,\mathsf{E}_2\}\,, \quad \frac{\partial}{\partial a}\,b\,\mathsf{E}_2 = \emptyset\,, \quad \frac{\partial}{\partial b}\,b\,\mathsf{E}_2 = \{\mathsf{E}_2\}\,,$$
$$\frac{\partial}{\partial a}\,[a\,\mathsf{H}_2\,\mathsf{E}_2] = \{\mathsf{H}_2\,\mathsf{E}_2\}\,, \quad \frac{\partial}{\partial b}\,[a\,\mathsf{H}_2\,\mathsf{E}_2] = \emptyset\,,$$
$$\frac{\partial}{\partial a}\,[\mathsf{H}_2\,\mathsf{E}_2] = \{b\,\mathsf{E}_2, a\,\mathsf{H}_2\,\mathsf{E}_2\}\,, \quad \frac{\partial}{\partial b}\,[\mathsf{H}_2\,\mathsf{E}_2] = \{\mathsf{H}_2\,\mathsf{E}_2\}\,.$$

Thus E_2 has 4 derived terms: E_2 itself, $b\,\mathsf{E}_2$, $a\,\mathsf{H}_2\,\mathsf{E}_2$ and $\mathsf{H}_2\,\mathsf{E}_2$.

Example 2.6. Let $\mathsf{E}_1 = a^* + a^*b(ba^*b)^*ba^* + a^*b(ba^*b)^*a(b+a(ba^*b)^*a)^*a(ba^*b)^*ba^*$ be the expression computed in Figure 1 (a). For conciseness we put:

$$\mathsf{F}_1 = (b\,a^*b)^*a\,, \qquad \mathsf{G}_1 = (b + a\,(b\,a^*b)^*a)^*a\,, \qquad \text{and} \qquad \mathsf{H}_1 = (b\,a^*b)^*b\,a^*\,.$$

It holds: $\mathsf{E}_1 = a^* + a^*b\,\mathsf{H}_1 + a^*b\,\mathsf{F}_1\,\mathsf{G}_1\,\mathsf{H}_1$, and the computation of the derived terms of E_1 goes as follow:

$$\frac{\partial}{\partial a}\,\mathsf{E}_1 = \{a^*,\ a^*b\,\mathsf{H}_1,\ a^*b\,\mathsf{F}_1\,\mathsf{G}_1\,\mathsf{H}_1\}\,, \quad \frac{\partial}{\partial b}\,\mathsf{E}_1 = \{\mathsf{H}_1,\ \mathsf{F}_1\,\mathsf{G}_1\,\mathsf{H}_1\}\,,$$
$$\frac{\partial}{\partial a}\,a^* = \{a^*\}\,, \quad \frac{\partial}{\partial b}\,a^* = \emptyset\,, \quad \frac{\partial}{\partial a}\,a^*b\,\mathsf{H}_1 = \{a^*b\mathsf{H}_1\}\,, \quad \frac{\partial}{\partial b}\,a^*b\,\mathsf{H}_1 = \{\mathsf{H}_1\}\,,$$
$$\frac{\partial}{\partial a}\,\mathsf{H}_1 = \emptyset\,, \quad \frac{\partial}{\partial b}\,\mathsf{H}_1 = \{a^*,\ a^*b\,\mathsf{H}_1\}\,,$$
$$\frac{\partial}{\partial a}\,a^*b\,\mathsf{F}_1\,\mathsf{G}_1\,\mathsf{H}_1 = \{a^*b\,\mathsf{F}_1\,\mathsf{G}_1\,\mathsf{H}_1\}\,, \quad \frac{\partial}{\partial b}\,a^*b\,\mathsf{F}_1\,\mathsf{G}_1\,\mathsf{H}_1 = \{\mathsf{F}_1\,\mathsf{G}_1\,\mathsf{H}_1\}\,,$$
$$\frac{\partial}{\partial a}\,\mathsf{F}_1\,\mathsf{G}_1\,\mathsf{H}_1 = \{\mathsf{G}_1\,\mathsf{H}_1\}\,, \quad \frac{\partial}{\partial b}\,\mathsf{F}_1\,\mathsf{G}_1\,\mathsf{H}_1 = \{a^*b\,\mathsf{F}_1\,\mathsf{G}_1\,\mathsf{H}_1\}\,,$$
$$\frac{\partial}{\partial a}\,\mathsf{G}_1\,\mathsf{H}_1 = \{\mathsf{F}_1\,\mathsf{G}_1\,\mathsf{H}_1,\ \mathsf{H}_1\}\,, \quad \frac{\partial}{\partial b}\,\mathsf{G}_1\,\mathsf{H}_1 = \{\mathsf{G}_1\,\mathsf{H}_1\}\,,$$

Thus E_1 has 7 derived terms: E_1 itself, a^* , $a^*b\,\mathsf{H}_1$, H_1 , $a^*b\,\mathsf{F}_1\,\mathsf{G}_1\,\mathsf{H}_1$, $\mathsf{F}_1\,\mathsf{G}_1\,\mathsf{H}_1$, and $\mathsf{G}_1\,\mathsf{H}_1$.

The above definition is the one given by Antimirov, which we have kept for accurate reference. We now slightly modify the definition of derived terms in order to reach our goal. In Antimirov's definition, an expression E always belongs to the set of its derived terms; in the new one, we get rid of this constraint, for what is needed is only the property that E be a union of derived terms. We thus define[6] a new operation on rational expressions which, roughly speaking, consists of decomposing an expression into a set of expressions whose left factor is not a sum.

[6]As explained in [12], this operation can be considered as a derivation with respect to the empty word.

Definition 2.7. i) The set of *initial derived terms* of an expression E is a set $d(\mathsf{E})$ of expressions inductively defined by:

$$d(0) = \{0\}\,, \qquad d(1) = \{1\}\,, \qquad d(a) = \{a\}, \forall a \in A$$
$$d(\mathsf{E}+\mathsf{F}) = d(\mathsf{E}) \cup d(\mathsf{F})\,, \qquad d(\mathsf{E}\cdot\mathsf{F}) = [d(\mathsf{E})]\cdot\mathsf{F}\,, \qquad d(\mathsf{E}^*) = \{\mathsf{E}^*\}.$$

ii) The *set of derived terms* of E is redefined as the smallest set that contains the initial derived terms of E and that is closed under derivation (in the sense of Definition 2.1).

From now on, "*derived terms*" is understood according to Definition 2.7.

Example 2.8 (*Ex. 2.5 cont.*)**.** As $d(\mathsf{E}_2) = \mathsf{E}_2$, the new definition does not change the derived terms of E_2.

Example 2.9 (*Ex. 2.6 cont.*)**.** It holds $d(\mathsf{E}_1) = \{a^*,\ a^*b\,\mathsf{H}_1,\ a^*b\,\mathsf{F}_1\,\mathsf{G}_1\,\mathsf{H}_1\}$ and E_1 has 6 derived terms: a^*, $a^*b\,\mathsf{H}_1$, H_1, $a^*b\,\mathsf{F}_1\,\mathsf{G}_1\,\mathsf{H}_1$, $\mathsf{F}_1\,\mathsf{G}_1\,\mathsf{H}_1$, and $\mathsf{G}_1\,\mathsf{H}_1$.

In [1], Antimirov has defined an automaton by means of the derived terms and we use here the same construction *mutatis mutandis.*[7]

Definition 2.10. The *derived term automaton* of a rational expression E is the finite automaton $\mathcal{A}_\mathsf{E}$ whose states are the derived terms of E and whose transitions are defined by:

i) if K and K' are derived terms of E and if a is a letter of A, $(\mathsf{K}, a, \mathsf{K}')$ is a transition of $\mathcal{A}_\mathsf{E}$ if and only if K' belongs to $\frac{\partial}{\partial a}\mathsf{K}$;

ii) the initial states[8] of $\mathcal{A}_\mathsf{E}$ are the initial derived terms of E;

iii) a derived term K is a final state of $\mathcal{A}_\mathsf{E}$ if and only if $\mathsf{c}(\mathsf{K}) = 1$;

The automaton $\mathcal{A}_\mathsf{E}$ recognizes the language denoted by E (the proof goes as in [1], or by the general argument quoted in the footnote). In the sequel, we denote by Δ the function that maps a rational expression onto its derived term automaton: $\Delta(\mathsf{E}) = \mathcal{A}_\mathsf{E}$ (and Δ is thus a Ψ-type algorithm).

Figure 2 and 3 show the derived term automaton of E_2 and E_1 respectively.

2.2. Automata morphisms, automata quotients

We describe now a process of *minimization* of automata which is a generalization of the classical minimization of deterministic automata (similar to the definition of *simulation* among *transition systems*). This notion applies to any kind of automata and is more versatile. On the other hand, this process yields an automaton which

[7] In the framework of power series, if U is a module of finite type closed under left quotient, the choice of any set of generators of U gives rise to an automaton that recognizes any series of U (modulo the adequate choice of initial coefficients) [15]. The automaton of derived terms is precisely the automaton corresponding to the generators denoted by the derived terms.

[8] The quality of being initial or final for a state is of the same kind as the definition of a transition: *cf.*below.

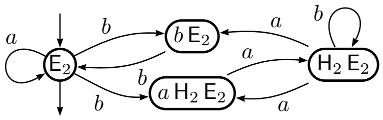


FIGURE 2. The derived term automaton of $\mathsf{E}_2 = (a + bb + ba(b + aa)^*ab)^*$

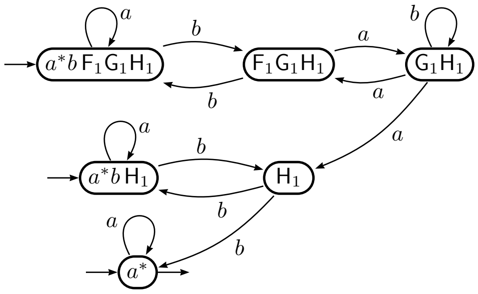


FIGURE 3. The derived term automaton of
$\mathsf{E}_1 = a^* + a^*b(ba^*b)^*ba^* + a^*b(ba^*b)^*a(b + a(ba^*b)^*a)^*a(ba^*b)^*ba^*$

is not canonically attached to the recognized language anymore but depends on the automaton it is computed from.

All this is based on the notion of *morphism of automata* and of their *local properties*. This is not new by far, and can be found in previous work of the authors (*cf.* [14]) or of others (*e.g.* [2]) with possibly different wording. For sake of completeness and in order to have the definitions we need under the precise statement we shall use, we briefly recall all that matter now. More details can be found in [15] for instance.

2.2.1. *Morphisms of automata*

Given an automaton $\mathcal{A} = \langle Q, A, E, I, T \rangle$, the set E of labelled edges is canonically equipped with three mappings (the three projections): $\iota\colon E \to Q$, $\tau\colon E \to Q$ and $\varepsilon\colon E \to A^*$, that is the state $e\iota$ is the origin of the transition e, $e\tau$ its end and $e\varepsilon$ its label.

A morphism φ from $\mathcal{A} = \langle Q, A, E, I, T \rangle$ into $\mathcal{B} = \langle R, A, F, J, U \rangle$ is indeed a *pair of mappings* (both denoted by φ): one between the set of states $\varphi\colon Q \to R$ and one between the set of transitions $\varphi\colon E \to F$ which satisfy the three properties:

$$\varphi \circ \iota = \iota \circ \varphi \quad \text{and} \quad \varphi \circ \tau = \tau \circ \varphi \,, \tag{7}$$

$$\varphi \circ \varepsilon = \varepsilon \,, \tag{8}$$

$$I\varphi \subseteq J \quad \text{and} \quad T\varphi \subseteq U \,. \tag{9}$$

Conditions (7) imply that *the image of a computation in $\mathcal{A}$ is a computation in $\mathcal{B}$.* Condition (8) implies that *the label of a computation in $\mathcal{A}$ is the same as the label of the image of that computation in $\mathcal{B}$.* Conditions (9) imply that *the image of a successful computation in $\mathcal{A}$ is a successful computation in $\mathcal{B}$.* In particular, if $\varphi\colon \mathcal{A} \to \mathcal{B}$ is a morphism, it holds $L(\mathcal{A}) \subseteq L(\mathcal{B})$.

The composition of two morphisms is a morphism.

Remark 2.11. Let us note that conditions (9) may adequately be replaced by the following convention. By way of a kind of implicit normalization, every automaton $\mathcal{A}$ is equipped with a "hidden" initial state $i_{\mathcal{A}}$ and a "hidden" final state $t_{\mathcal{A}}$. Then every initial state i of $\mathcal{A}$ becomes only a state such that there exists a transition $(i_{\mathcal{A}}, 1_A^*, i)$ from $i_{\mathcal{A}}$ to i labelled by the empty word. Similarly, every final state t of $\mathcal{A}$ becomes a state such that there exists a transition $(t, 1_A^*, t_{\mathcal{A}})$. Obviously, if $\varphi\colon \mathcal{A} \to \mathcal{B}$ is a morphism of automata, φ maps $i_{\mathcal{A}}$ onto $i_{\mathcal{B}}$, $t_{\mathcal{A}}$ onto $t_{\mathcal{B}}$ and then (9) is implied by (7) and (8).

2.2.2. *Local properties of morphisms*

For every state q of an automaton $\mathcal{A} = \langle Q, A, E, I, T\rangle$, let us denote by $\mathsf{Out}_{\mathcal{A}}(q)$ the set of edges of $\mathcal{A}$ *the origin of which* is q, that is edges that are "going out" of q :

$$\mathsf{Out}_{\mathcal{A}}(q) = \{e \in E \mid e\iota = q\} .$$

Dually, $\mathsf{In}_{\mathcal{A}}(q)$ is the set of edges of $\mathcal{A}$ *the end of which* is q, that is edges that are "going in" q :

$$\mathsf{In}_{\mathcal{A}}(q) = \{e \in E \mid e\tau = q\} .$$

If φ is a morphism from $\mathcal{A} = \langle Q, A, E, I, T\rangle$ into $\mathcal{B} = \langle R, A, F, J, U\rangle$ then for every p in Q, φ maps $\mathsf{Out}_{\mathcal{A}}(p)$ into $\mathsf{Out}_{\mathcal{B}}(p\varphi)$, and $\mathsf{In}_{\mathcal{A}}(p)$ into $\mathsf{In}_{\mathcal{B}}(p\varphi)$.

We say that φ is *Out-surjective* if for every p in Q the restriction of φ to $\mathsf{Out}_{\mathcal{A}}(p)$ is surjective onto $\mathsf{Out}_{\mathcal{B}}(p\varphi)$. Accordingly, we say that φ is *In-surjective* if for every p in Q the restriction of φ to $\mathsf{In}_{\mathcal{A}}(p)$ is surjective onto $\mathsf{In}_{\mathcal{B}}(p\varphi)$

With the convention of Remark 2.11, it follows that if φ is Out-surjective then $U\varphi^{-1} \subseteq T$ and thus, by (9), $U\varphi^{-1} = T$; moreover, if j is an initial state in $\mathcal{B}$, then there exists at least one initial state of $\mathcal{A}$ in $j\varphi^{-1}$. Dually, if φ is In-surjective, then $J\varphi^{-1} = I$ and if u is a final state in $\mathcal{B}$, then there exists at least one final state of $\mathcal{A}$ in $u\varphi^{-1}$.

It is clear that the composition of two Out-surjective (resp. In-surjective) morphisms is an Out-surjective (resp. In-surjective) morphism. The following proposition is easily established by induction on the length of the computations.

Proposition 2.12. *Let $\varphi\colon \mathcal{A} \to \mathcal{B}$ be an Out-surjective (resp. In-surjective) morphism. For every computation c in $\mathcal{B}$ whose origin (resp. whose end) r is in the image of φ* and for every p of Q such $p\varphi = r$, *there exists at least one computation d in $\mathcal{A}$ whose origin (resp. whose end) is p and such that $d\varphi = c$.*

In particular if $\varphi\colon \mathcal{A} \to \mathcal{B}$ is Out-surjective, $L(\mathcal{A}) = L(\mathcal{B})$ holds.

2.2.3. *Quotients of automata*

We say that $\varphi\colon \mathcal{A} \to \mathcal{B}$ is *globally*[9] surjective if $Q\varphi = R$ and $E\varphi = F$. The morphism φ will be said *totally surjective* (resp. *totally co-surjective*) if it is globally surjective and Out-surjective (resp. In-surjective).

Definition 2.13. An automaton $\mathcal{B}$ is a *quotient* (resp. a *co-quotient*) of $\mathcal{A}$ if there exists a morphism $\varphi\colon \mathcal{A} \to \mathcal{B}$ that is totally surjective (resp. totally co-surjective).

Proposition 2.14. *Every automaton $\mathcal{A}$ has a* mimimal quotient $\mathcal{C}$, *which is unique up to an isomorphism, and which is the quotient of any quotient $\mathcal{B}$ of $\mathcal{A}$. More precisely, there exists a totally surjective morphism $\psi\colon \mathcal{A} \to \mathcal{C}$ such that for any totally surjective morphism $\varphi\colon \mathcal{A} \to \mathcal{B}$ there exists a totally surjective morphism $\theta\colon \mathcal{B} \to \mathcal{C}$ such that $\psi = \varphi\,\theta$.*

Obviously, the dual proposition holds:

Proposition 2.15. *Every automaton $\mathcal{A}$ has a* mimimal co-quotient $\mathcal{D}$, *which is unique up to an isomorphism, and which is the co-quotient of any co-quotient $\mathcal{B}$ of $\mathcal{A}$.*

Remark 2.16. In the terminology of transition systems [2], $\mathcal{A}$ and $\mathcal{C}$ are *in bisimulation* if they have the same minimal quotient.

The minimal quotient or co-quotient of an automaton can be computed by a kind of Moore algorithm that consists in successive refinements of the trivial partition on the set of states. As we shall be more interested in the sequel by the co-quotient than by the quotient, we rather describe the algorithm for the co-quotient.

Let $\mathcal{A} = \langle Q, A, E, I, T\rangle$ be an automaton. For every letter a in A and every state p in Q, the set of *predecessors* of p by a is denoted by $a\cdot p = \{q \mid (q, a, p) \in E\}$. We define the sequence $\{\mathcal{Q}_0, \mathcal{Q}_1, \ldots\}$ of partitions of Q ordered by inclusion, by the following rules:

- $\mathcal{Q}_0 = \{Q \setminus I, I\}$;
- For every k, $\mathcal{Q}_{k+1}$ is *the* refinement of $\mathcal{Q}_k$ such that, for every states p and q in the same class in $\mathcal{Q}_k$, p and q are in the same class in $\mathcal{Q}_{k+1}$ if and only if, for every letter a in A, $a \cdot p$ and $a \cdot q$ intersect the same classes of $\mathcal{Q}_k$.

The algorithm stops when $\mathcal{Q}_k$ cannot be refined by the preceeding rules. Let us denote by l this last index, if it exists. If Q is finite, l exists and is smaller than $\mathsf{Card}(Q)$.

Let us denote by ψ_k the (surjective) canonical mapping from Q onto the classes of $\mathcal{Q}_k$. The refinement from $\mathcal{Q}_k$ to $\mathcal{Q}_{k+1}$ is strict iff ψ_k is not In-surjective thus ψ_l is indeed an In-surjective morphism (and thus totally co-surjective).

We shall say that two states p and q which belong to the same class of $\mathcal{Q}_l$ (*i.e.* that have the same image by ψ_l) are *In-similar*.

[9] Out-surjectivity (or In-surjectivity) is a "local" condition, which has to be verified at every state of the automaton.

Theorem 2.17. *The automaton* $\mathcal{C} = \mathcal{A}/\psi_l$ *computed by the above Moore algorithm is the minimal co-quotient of* $\mathcal{A}$*.*

We denote this co-minimization algorithm by Υ. Let us note again that $\Upsilon(\mathcal{A})$ is canonically attached to $\mathcal{A}$ and not to the language accepted by $\mathcal{A}$, but for the case where $\mathcal{A}$ is co-deterministic, in which case $\Upsilon(\mathcal{A})$ is *the* minimal co-deterministic automaton of $L(\mathcal{A})$

Example 2.18 (*Ex. 2.5 cont.*)**.** In the automaton of Figure 2, the states $b\,\mathsf{E}_2$ and $a\,\mathsf{H}_2\,\mathsf{E}_2$ are In-similar. The minimal co-quotient is therefore $\mathcal{P}_1$ (Figure 1).

Example 2.19 (*Ex. 2.6 cont.*)**.** On Figure 3, the states a^*, $a^*b\mathsf{H}_1$ and $a^*b\mathsf{F}_1\mathsf{G}_1\mathsf{H}_1$ on the one hand, and the states H_1 and $\mathsf{F}_1\mathsf{G}_1\mathsf{H}_1$ on the other hand, are In-similar. Hence, the minimal co-quotient of $\Delta(\mathsf{E}_1)$ is the automaton $\mathcal{P}_1$ again.

3. Building a solution

The last two examples show two instances where:

$$\mathcal{A} = \Upsilon \circ \Delta \circ \Phi(\mathcal{A}) \tag{10}$$

and this is the main idea of the paper: $\Upsilon \circ \Delta$ is "fundamentally" the inverse of Φ (the state elmination method).

3.1. The core of the algorithm and its shortcoming

We develop in this part some other examples where $\Upsilon \circ \Delta$ is the inverse of Φ and we explain then the obstacles that prevent this property to hold in the general case.

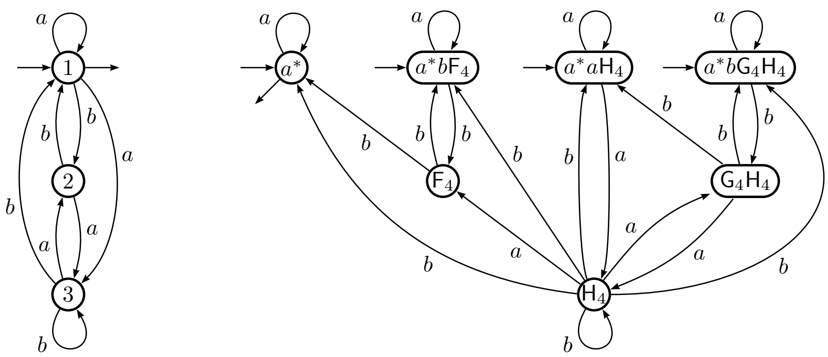
Example 3.1. Let us consider the automaton $\mathcal{A}_1$ of Figure 4 (a) and let E_4 be the expression obtained by the state elimination method on $\mathcal{A}_1$ using the order 1-2-3. It holds:

$$\begin{aligned} \mathsf{E}_4 &= a^* + a^*b\mathsf{F}_4 + \left(a^*a + a^*b\mathsf{G}_4\right)\mathsf{H}_4\,, \qquad \text{where} \\ \mathsf{F}_4 &= (ba^*b)^*ba^*\,, \quad \mathsf{G}_4 = (ba^*b)^*(a + ba^*a)\,, \\ \text{and} \quad \mathsf{H}_4 &= \left(b + ba^*a + (a + ba^*b)\mathsf{G}_4\right)^* \left(ba^* + (a + ba^*b)\mathsf{F}_4\right)\,. \end{aligned}$$

Then, $d(\mathsf{E}_4) = \{a^*,\ a^*b(ba^*b)^*ba^*,\ a^*a\mathsf{H}_4,\ a^*b(ba^*b)^*(a + ba^*a)\mathsf{H}_4\}$. And the derived terms of E_4 are read on the automaton $\Delta(\mathsf{E}_4)$ itself (Figure 4 (b)).

It is quite easy to check that the co-quotient of the automaton $\Delta(\mathsf{E}_4)$ is the automaton $\mathcal{A}_1$ itself. Actually, the morphism that maps every state of $\Delta(\mathsf{E}_4)$ onto the state of $\mathcal{A}_1$ drawn on the same horizontal line is In-surjective.

From the automaton $\mathcal{A}_1$, another rational expression can be computed, by running the algorithm Δ with another order, for instance 1-3-2. Let E_5 be the



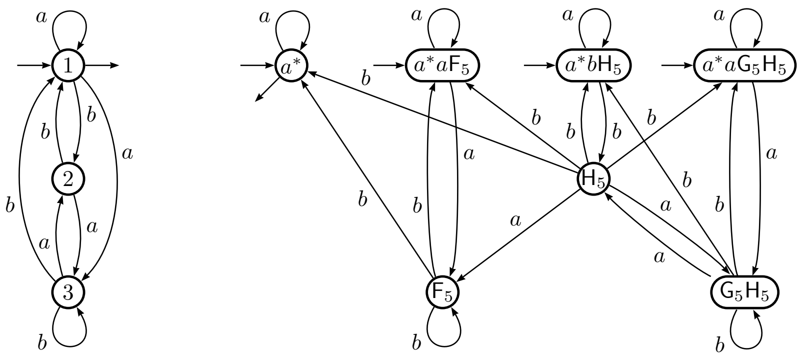
(a) An automaton $\mathcal{A}_1$ (b) The automaton $\Delta(\mathsf{E}_4)$ of derived terms of E_4

FIGURE 4. The core of the algorithm applied to $\mathcal{A}_1$

resulting expression:

$$\begin{aligned} \mathsf{E}_5 &= a^* + a^*a\mathsf{F}_5 + \big(a^*b + a^*a\mathsf{G}_5\big)\,\mathsf{H}_5\,, \qquad \text{where} \\ \mathsf{F}_5 &= (b+ba^*a)^*ba^*\,, \quad \mathsf{G}_5 = (b+ba^*a)^*(a+ba^*b)\,, \\ \text{and}\quad \mathsf{H}_5 &= \big(ba^*b + (a+ba^*a)\mathsf{G}_5\big)^*\,\big(ba^* + (a+ba^*a)\mathsf{F}_5\big)\ . \end{aligned}$$

Figure 5 (b) shows the derived term automaton of E_5. Once again, the co-quotient of the automaton $\Delta(\mathsf{E}_5)$ is the automaton $\mathcal{A}_1$.



(a) The automaton $\mathcal{A}_1$ (b) The automaton $\Delta(\mathsf{E}_5)$ of derived terms of E_5

FIGURE 5. The same algorithm and another ordering

This example shows two instances where, again:

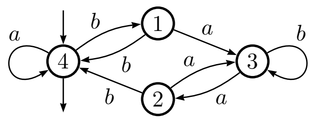
$$\mathcal{A} = \Upsilon \circ \Delta \circ \Phi(\mathcal{A}), \tag{11}$$

and this with an automaton which is neither deterministic nor co-deterministic. Observe that this would not hold if we had not modified the definition of the derivation: in Example 3.1, with the Antimirov's definition of the derivation, $\Delta(\mathsf{E}_4)$ whould have one state more and would not accept $\mathcal{A}_1$ as co-quotient.

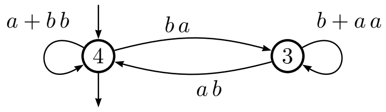
However, it is clear that (11) cannot hold in full generality: if $\mathcal{A}$ is not co-minimal for instance, certainly (11) does not hold; $\mathcal{A}$ may then be a quotient of $\Delta \circ \Phi(\mathcal{A})$, but one does not know how to chose which one.

But the situation is even more tricky and it may happen that (11) does not hold even for co-minimal automata, as shown by the following example.

Example 3.2. The automaton $\mathcal{P}_1'$ (Figure 6) is co-minimal. After two steps of the elimination Φ, the configuration (Figure 6 (b)) is the same as the one obtained after one step of the elimination on the automaton of Figure 1 (b). Thus $\Phi(\mathcal{P}_1') = \mathsf{E}_2$ and $\Upsilon(\Delta(\mathsf{E}_2))$ is equal to $\mathcal{P}_1$ and not to $\mathcal{P}_1'$.



(a) The automaton $\mathcal{P}_1'$



(b) $\mathcal{P}_1'$ after the elimination of two states

FIGURE 6. The state elimination method on the automaton $\mathcal{P}_1'$

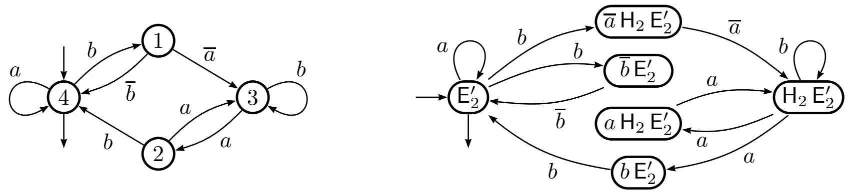
3.2. TAGGING FOR A SOLUTION

A way of escaping the above mentioned difficulties is to "decorate" some labels of the automaton in order to indicate in the expression that some occurences of letters in the expression come from different transitions. We call this operation a *partial linearization* and we denoted it by Λ. The delinearization is a projection that we denote by Π.

Example 3.3 (*Ex. 3.2 cont.*)**.** $\mathcal{P}_1'$ is linearized into $\mathcal{P}_1''$ as shown on Figure 7; the result of Φ is $\mathsf{E}_2' = (a + b\overline{b} + b\overline{a}(b + aa)^*ab)^*$, and $\Delta(\mathsf{E}_2')$ is an automaton whose minimal co-quotient is $\mathcal{P}_1''$.

The linearization may consists in tagging not only letters on the transitions of the automaton but spontaneous transitions from a final state to the "hidden" final state as shown in the following example.[10]

[10]Obviously, the tagging of the spontaneous transitions from the "hidden" initial state to an initial state is possible, but it will appear later (Theorem 3.5) that this is useless.



(a) The linearized automaton $\mathcal{P}_1''$

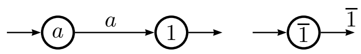
(b) The derived term automaton of E_2'.

FIGURE 7. The complete algorithm on automaton $\mathcal{P}_1'$

Example 3.4. Let $\mathcal{A}_2$ be the automaton of Figure 8 (a). Without any tag, the expression computed from $\mathcal{A}_2$ is $\Phi(\mathcal{A}_2) = 1 + a$. If we apply Δ, we obtain the automaton of Figure 8 (b), which is co-minimal. But, if one of the terminal arrows of $\mathcal{A}_2$ is tagged, the expression obtained is $\overline{1} + a$ and the automaton of derived terms is the automaton of Figure 8 (c) whose minimal co-quotient is $\mathcal{A}_2$.

(a) The minimal automaton $\mathcal{A}_2$

(b) The co-minimal automaton



(c) The automaton $\Delta(\Phi(\Lambda(\mathcal{A}_2)))$

FIGURE 8. Three automata that recognize $1 + a$

This tagging of final states has to appear in the rational expression computed from the automaton and is symbolized by tagged 1's. Like the transitions labelled by letters, the spontaneous transitions to the hidden final state keep their tagging in the derived term automaton.

The aim is obviously to keep the linearization as small as possible. Indeed, if the linearization makes every arrow of $\mathcal{A}$ distinct, the expression is nothing else than a description of the automaton. Fortunately, the linearization required to retrieve the automaton is far more succint. Unfortunately, as far as now, we are not able to give a characterization of the necessary and sufficient linearization. We give in the following subsection a sufficient condition which is certainly not a necessary one.

3.3. A sufficient tagging

This condition allows somehow to get around the difficulty of choosing a good linearization. It does not take advantage of the properties of the minimal co-quotient of an automaton but refer to the language accepted by the automaton.

We shall establish that $\Delta \circ \Phi$ preserves the co-determinism and this will imply that a tagging which makes the automaton co-deterministic is certainly sufficient.

Theorem 3.5. *Let $\mathcal{A}$ be a co-deterministic automaton and $\mathsf{E} = \Phi(\mathcal{A})$ a rational expression computed from $\mathcal{A}$ by the state elimination method. Then, the derived term automaton $\Delta(\mathsf{E})$ of E is co-deterministic.*

As a direct consequence of Theorem 3.5, we have:

Theorem 3.6. *Let $\mathcal{A}$ be an automaton, Λ a partial linearization that makes $\Lambda(\mathcal{A})$ a minimal co-deterministic automaton and Π the corresponding delinearization. It then holds:*

$$\mathcal{A} = \Pi \circ \Upsilon \circ \Delta \circ \Phi \circ \Lambda(\mathcal{A})\,. \tag{12}$$

Proof. The minimal co-quotient of any co-deterministic automaton is the co-minimal automaton of the language. Therefore, if $\Lambda(\mathcal{A})$ is the co-minimal automaton of the language denoted by $\mathsf{E} = \Phi(\Lambda(\mathcal{A}))$, $\Upsilon(\Delta(\mathsf{E})) = \Lambda(\mathcal{A})$ and $\mathcal{A} = \Pi \circ \Lambda(\mathcal{A})$. □

If we come back to the notation of the introduction, Theorem 3.6 gives the Φ' and Ω we are looking for: $\Phi' = \Phi \circ \Lambda$ and $\Omega = \Pi \circ \Upsilon \circ \Delta$.

It may seem that the hypothesis on Λ makes Theorem 3.6 obvious since, as soon as it is known that $\mathcal{B}$ is a minimal co-deterministic automaton, it is easy to retrieve it from any description of the language $L(\mathcal{B})$. This is not the case because the algorithm described in this theorem is not a classical one to compute a minimal automaton and in particular involves no determinization procedure.

Another way to understand the true meaning of Theorem 3.5 (and thus of Theorem 3.6) is to state the result in the dual form:

Corollary 3.7. *Let E be a rational expression computed from a* deterministic *automaton $\mathcal{A}$. Then, the automaton of* right *derived terms of E is deterministic.*

Right derived terms are obtained by replacing the derivation (which is a *left* derivation) by a *right* derivation $\dfrac{\partial_{\mathrm{R}}}{\partial a}$, defined by the same induction rules as in Definition 2.1 but for (4) and (5) which are replaced by:

$$\frac{\partial_{\mathrm{R}}}{\partial a}(\mathsf{E}\cdot\mathsf{F}) = \mathsf{E}\cdot\left[\frac{\partial_{\mathrm{R}}}{\partial a}\,\mathsf{F}\right] \cup \mathsf{c}(\mathsf{F})\,\frac{\partial_{\mathrm{R}}}{\partial a}\,\mathsf{E} \tag{4'}$$

$$\frac{\partial_{\mathrm{R}}}{\partial a}(\mathsf{E}^*) = \mathsf{E}^*\cdot\left[\frac{\partial_{\mathrm{R}}}{\partial a}\,\mathsf{E}\right] \tag{5'}$$

The definition of the *automaton of right derived terms* is dual to the one of the automaton of derived terms: $(\mathsf{K}, a, \mathsf{K}')$ is a transition if K belongs to $\frac{\partial_{\mathrm{R}}}{\partial a}\mathsf{K}'$ and the definition of initial and final states is changed accordingly. This deterministic automaton has a linear number of states (in the size of E) and its computation does not call any determinization algorithm. To the authors' knowledge there exists no other algorithm with the same properties in the literature.

Proof of Theorem 3.5. Let $\mathcal{A}$ be a co-deterministic automaton and let E be the expression computed by the state elimination method with respect to an order ω.

For every expression F, we define the sets of occurences of letters in F, $\mathrm{First}(\mathsf{F})$ and $\mathrm{Last}(\mathsf{F})$ as in the algorithm which computes the position automaton of F (*cf.* [11]).

$$\begin{aligned}
\mathrm{First}(\mathsf{0}) &= \mathrm{First}(\mathsf{1}) = \emptyset & \mathrm{Last}(\mathsf{0}) &= \mathrm{Last}(\mathsf{1}) = \emptyset \\
\mathrm{First}(a) &= a & \mathrm{Last}(a) &= a \\
\mathrm{First}(\mathsf{E}+\mathsf{F}) &= \mathrm{First}(\mathsf{E}) \cup \mathrm{First}(\mathsf{F}) & \mathrm{Last}(\mathsf{E}+\mathsf{F}) &= \mathrm{Last}(\mathsf{E}) \cup \mathrm{Last}(\mathsf{F}) \\
\mathrm{First}(\mathsf{E}\cdot\mathsf{F}) &= \mathrm{First}(\mathsf{E}) \cup \mathsf{c}(\mathsf{E})\mathrm{First}(\mathsf{F}) & \mathrm{Last}(\mathsf{E}\cdot\mathsf{F}) &= \mathrm{Last}(\mathsf{F}) \cup \mathsf{c}(\mathsf{F})\mathrm{Last}(\mathsf{E}) \\
\mathrm{First}(\mathsf{E}^*) &= \mathrm{First}(\mathsf{E}) & \mathrm{Last}(\mathsf{E}^*) &= \mathrm{Last}(\mathsf{E})
\end{aligned}$$

It can be shown by induction that, for every derived term K of F, $\mathrm{First}(\mathsf{K})$ is in bijection with the transitions that go out of K in the derived term automaton of F.

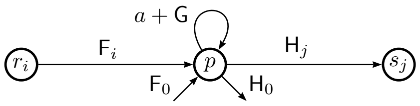
We can first notice the following fact (that is easy to prove by induction): in the course of the elimination state method, for every state p, for every incoming edge on p (incoming transition or initial edge) labelled by F, no element of $\mathrm{Last}(\mathsf{F})$ is under a star operator. More, if F is not a letter, the element of $\mathrm{Last}(\mathsf{F})$ come from the incoming transitions on p that have been deleted before. Likewise, $\mathsf{c}(\mathsf{F})$ is false.

Let us focus on a transition (p, a, q) of $\mathcal{A}$ and study the position of occurences of *this* a in E (*i.e.* that comes from this transition). It depends on the order between p and q.

Our aim is to described the form of derived terms K such that this a belongs to $\mathrm{First}(\mathsf{K})$. We want to prove that the derived terms that are obtained by the derivation of this occurence give states that are (locally) co-deterministic with respect to a.

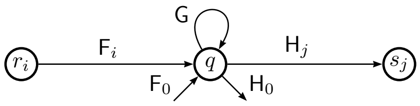
If we prove this fact for every transition, then every state of the automaton of derived term is locally co-deterministic with respect to every letter. Hence, the automaton of derived terms is co-deterministic.

Case 1: $p = q$. Before the elimination of p, the local configuration of the automaton is:



In this case, this a may appear several times in E, but each of its occurences belong to a factor of the form $\mathsf{F}_i(a+\mathsf{G})^*\mathsf{H}_j$. For every i, First($\mathsf{F}_i(a+\mathsf{G})^*\mathsf{H}_j$)=First($\mathsf{F}_i$) and a is neither in Last(F_i) nor in Last(G), since $\mathcal{A}$ is co-deterministic. Therefore the only derived terms such that this occurence of a belongs to the First set, are the derived terms that begins with $(a+\mathsf{G})^*\mathsf{H}_j$. The only way to reach one of these derived terms K from another one is by derivating F_i or G and the resulting transitions are labelled by a letter different from a. The derivative of K with respect to this occurence of a is K itself. Notice that a may belong to First(H_j), but this occurence comes from another transition (an outgoing transition from p) and we shall deal with this configuration in Case 3.

Case 2: $p < q$. The state p is eliminated before the state q. Before the elimination of q, the local configuration of the automaton is:



a may belong to the Last set of F_i and G. There are two subcases:
– when p has been eliminated, it had a loop labelled by $\mathsf{G}^{(p)}$. Then, every occurence of a is preceded by $(\mathsf{G}^{(p)})^*$ and therefore, every occurence of a belongs to a factor $(\ldots(\mathsf{G}^{(p)})^*a)(\mathsf{G})^*\mathsf{H}_j$. Hence, for every derived term $\mathsf{K} = (\mathsf{G})^*\mathsf{H}_j\mathsf{K}'$, there is only one incoming transition labelled by a; it comes from the derived term $(\mathsf{G}^{(p)})^*a)(\mathsf{G})^*\mathsf{H}_j\mathsf{K}'$.
– if p had no loop, every occurence of a is preceded by a factor $\mathsf{F}_l^{(p)}$ (that labelled an incoming transition on p before its elimination) and no element of the Last sets of these factors is under a star; hence, for every derived term $\mathsf{K} = (\mathsf{G})^*\mathsf{H}_j\mathsf{K}'$, there is only one incoming transition labelled by a, which comes from the derived term $a(\mathsf{G})^*\mathsf{H}_j\mathsf{K}'$.

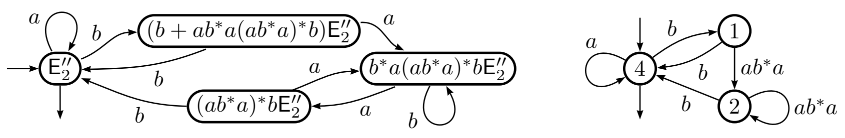
Case 3: $p > q$. The state q is eliminated first. Every occurence of this a in E belongs to a factor $a\mathsf{G}^*\mathsf{H}_j$ — if q has a loop labelled by G — or to a factor $a\mathsf{H}_j$ otherwise. Moreover, and in the latter case, no element of First(H_j) is under a star. Hence, for every derived term $\mathsf{K} = (\mathsf{G}^{(q)})^*\mathsf{H}_j^{(q)}\mathsf{K}'$ (*resp.* $\mathsf{K} = \mathsf{H}_j^{(q)}\mathsf{K}'$), there is only one incoming transition labelled by a, which comes from the derived term $a(\mathsf{G})^*\mathsf{H}_j\mathsf{K}'$ (*resp.* $a\mathsf{H}_j^{(q)}\mathsf{K}'$). □

4. Discussion

As we have already said, the conditions put on Λ in Theorem 3.6 are sufficient but far from being necessary. For instance, the automaton $\mathcal{A}_1$ in Example 3.1 is not co-deterministic and $\mathcal{A}_1 = \Upsilon(\Delta(\Phi(\mathcal{A}_1)))$ holds though. This example and other computations have led us to consider several ways that can help in distinguishing derived terms and thus reducing the role of Λ. As far as now, they can serve as heuristics and it is our current work to turn these ideas into precise statements. Let us quote here three of these directions of research.

Choosing a smart ordering. Actually, the choice of the ordering ω used in Φ may be very important. The automaton $\mathcal{P}'_1$ of Example 3.2 may give an illustration of this idea:

Example 4.1 (*Ex. 3.2 cont.*)**.** Let $\mathsf{E}''_2 = a + b(b + ab^*a(ab^*a)b)^*$ be the rational expression obtained by running Φ on $\mathcal{P}'_1$ (Figure 6) with order 3-2-4-1. Then, without any linearization, we get the derived term automaton of Figure 9 (a), whose minimal co-quotient is $\mathcal{P}'_1$.



(a) The automaton $\Delta(\mathsf{E}''_2)$ (b) The first step of the elimination

FIGURE 9. Running the algorithm with a smart ordering

One can try to explain why this ordering is relevant for our purpose. The elimination of the state 3 generates a loop on state 2 (Figure 9 (b)), and the label on that loop makes state 2 different from state 1 in the remaining of the computation. The choice of an ordering that builds such tagging loops during the elimination algorithm seems to be a good strategy to get an expression that is faithful to the automaton without linearization.

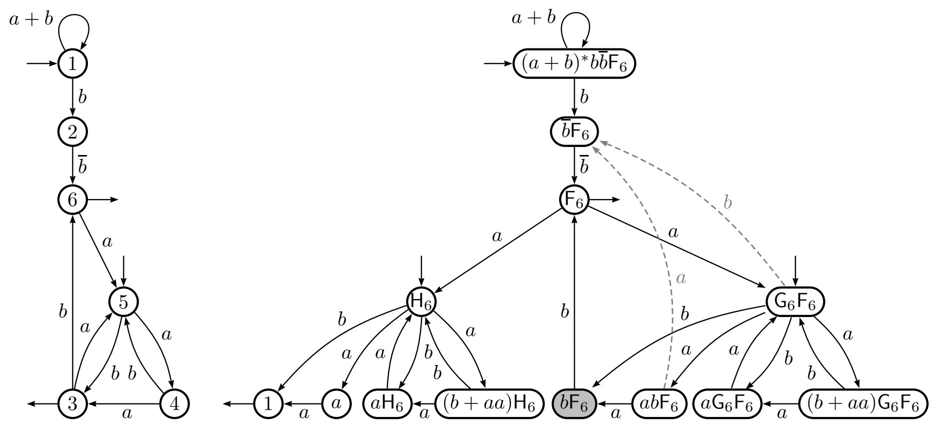
Using the structure of $\Delta(\Phi(\mathsf{E}))$. The previous examples show the remarkable structure of the derived term automaton of a rational expression computed on a *strongly connected* automaton $\mathcal{A}$: the last p of $\mathcal{A}$ to be eliminated corresponds to a term that is a *cutvertex* in $\Delta(\mathsf{E})$ and this property holds inductively on subautomata. This observation is another way to distinguish states that are otherwise labelled by a same derived term. It thus lead to an improved version of Δ which may again depend on the ordering of the elimination.

Example 4.2 (*Ex. 3.1 cont.*)**.** On Figure 4 (b), the state H_4 is the cutvertex. Actually, it corresponds to the state 3 of the automaton $\mathcal{A}_1$ that is eliminated last in Δ. On Figure 5, the state 2 of $\mathcal{A}_1$ is eliminated last and the corresponding state in $\Delta(\mathsf{E}_5)$ is indeed the cutvertex.

Example 4.3. Let $\mathcal{A}_6$ be the (linearized) automaton of Figure 10 (a). The elimination method with the ordering indicated by labels of states gives:

$$\mathsf{E}_6 = \Big((a+b)^*b\bar{b} + \mathsf{H}_6 b\Big)\,\mathsf{F}_6 + \mathsf{H}_6, \text{ where}$$
$$\mathsf{H}_6 = \big(ba + a(b+aa)\big)^*\,(b+aa) \quad\text{and}\quad \mathsf{F}_6 = (a\mathsf{H}_6 b)^*(1 + a\mathsf{H}_6).$$

The automaton $\Delta(\mathsf{E}_6)$ is drawn of Figure 10 (b), where $\mathsf{G}_6 = \mathsf{H}_6 b \mathsf{F}_6$. Without any linearization, the gray state $b\mathsf{F}_6$ is not built and its incomming transition are replaced by the dashed ones. In this case, no cutvertex can be found in $\Delta(\mathsf{E}_6)$; therefore, there are some derived terms that represent several states of the automaton that one tries to retrieve. The difficulty is obviously to decide which derived term has to be split to get a cutvertex...



(a) The automaton $\mathcal{A}_6$ (b) The automaton $\Delta(\mathsf{E}_6)$

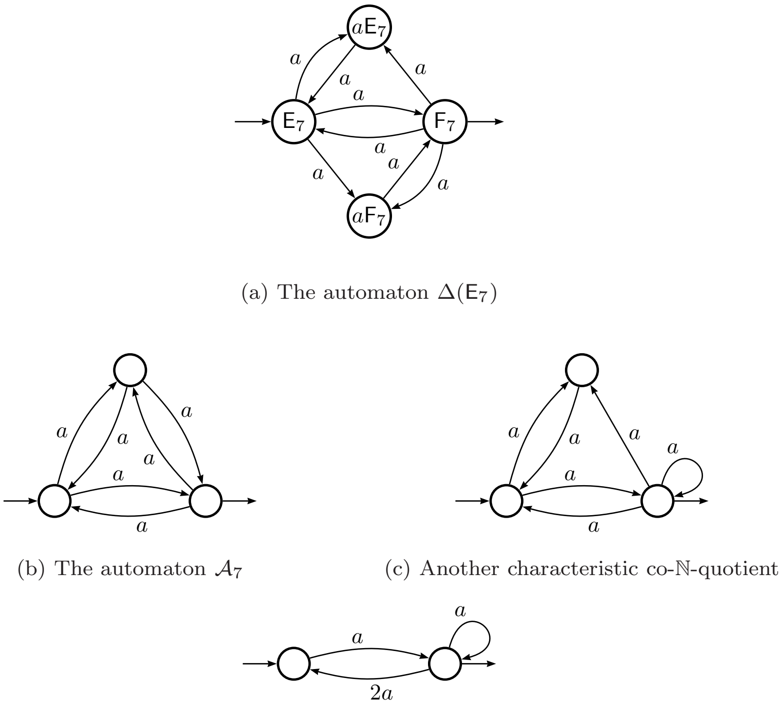
FIGURE 10. Using the structure of the automaton

Taking multiplicity into account. A fundamental property of the algorithms Φ and Δ is the fact that they respect the multiplicity of pathes. This is not necessarily the case for the co-quotient, if it is computed in a Boolean framework. To take this into account, we can compute co-quotient of automata with multiplicity, that means that every letter on transitions has a weight which is a (positive) integer. Such an automaton is a $\mathbb{N}$-automaton and for this kind of automaton, the notion of co-quotient can be generalized.

Definition 4.4. Let $\mathcal{A} = \langle Q, A, E, I, T\rangle$ and $\mathcal{B} = \langle R, A, F, J, U\rangle$ be two automata. Then, $\mathcal{B}$ is a *co-$\mathbb{N}$-quotient* of $\mathcal{A}$ if there exists a morphism μ from $\mathcal{A}$ onto $\mathcal{B}$ such that:

$$\forall q \in R, \quad \forall s \in q\mu^{-1}, \quad J_q = I_s, \qquad \forall p \in R, \quad U_p = \sum_{r \in p\mu^{-1}} T_r,$$
$$\forall p, q \in R, \quad \forall s \in q\mu^{-1}, \quad F_{p,q} = \sum_{r \in p\mu^{-1}} E_{r,s}.$$

Now, if we associate to the derived term automaton of an expression the corresponding characteristic automaton, that is the $\mathbb{N}$-automaton with the same transitions and with every weight equal to 1, a way to assure that the co-quotient does not fold some pathes is to assure that the co-ℕ-quotient does not produce any transition with weight greater than 1.



(a) The automaton $\Delta(\mathsf{E}_7)$

(b) The automaton $\mathcal{A}_7$

(c) Another characteristic co-ℕ-quotient

(d) The minimal co-ℕ-quotient of $\Delta(\mathsf{E}_7)$

FIGURE 11. Taking multiplicities into account

Example 4.5. Let E_7 be the expression computed from the automaton $\mathcal{A}_7$ of Figure 11 (b). $\mathsf{E}_7 = (aa)^*(a+aa)\mathsf{F}_7$, with $\mathsf{F}_7 = \big(aa + (a+aa)(aa)^*(a+aa)\big)^*$.

The derived terms automaton of E_7 has four states: E_7 itself, $a\mathsf{E}_7$, F_7 and $a\mathsf{F}_7$.

The minimal co-quotient of this automaton has two states, but it is not characteristic. Here arises a problem: there may be *several* smallest characteristic co-ℕ-quotients of an automaton. For instance, $\mathcal{A}_7$ is indeed a minimal characteristic co-ℕ-quotient of $\Delta(\mathsf{E}_7)$, but it is not unique; the automaton of Figure 11 (c) fills the same property. Taking multiplicities into account is here not sufficient to decide which co-quotient of $\Delta(\mathsf{E}_7)$ is $\mathcal{A}_7$.

5. Conclusion

In spite of the work that remains to be done on the subject, we hope that we have given strong evidences that the computation of the derived terms of an expression is not only an algorithm that builds an equivalent automaton but also a way to retrieve the "track" of the states of an automaton when the expression has been computed from that automaton. How far these tracks are faithful, and how to read them efficiently are questions that are still under investigation.

Acknowledgements The authors are pleased to thanks the careful referees who pointed out several inaccuracies in the definitions, and whose remarks were of precious help for improving the presentation of the paper.

References

[1] V. Antimirov, Partial derivatives of regular expressions and finite automaton constructions. *Theoret. Computer Sci.* **155** (1996), 291–319.
[2] A. Arnold, *Systèmes de transitions finis et sémantique des processus communiquants*, Masson, 1992. English Trans.: *Finite transitions systems*, Prentice-Hall, 1994.
[3] G. Berry et R. Sethi, From regular expressions to deterministic automata, *Theoret. Computer Sci.* **48** (1986), 117–126.
[4] J. Berstel et J.-E. Pin, Local languages and the Berry-Sethi algorithm, *Theoret. Computer Sci.* **155** (1996), 439–446.
[5] A. Brügemann-Klein, Regular expressions into finite automata. *Theoret. Comput. Sci.* **120** (1993), 197–213.
[6] J. A. Brzozowski, Derivatives of regular expressions. *J. Assoc. Comput. Mach.* **11** (1964), 481–494.
[7] P. Caron and D. Ziadi Characterization of Glushkov automata, *Theoret. Computer Sci.* **233** (2000), 75–90.
[8] J.-M. Champarnaud and D. Ziadi, New finite automaton constructions based on canonical derivatives. *Pre-Proceedings of CIAA'00*, M. Daley, M. Eramian and S. Yu, eds, Univ. of Western Ontario, (2000), 36–43.
[9] J.-M. Champarnaud and D. Ziadi, Canonical derivatives, partial derivatives and finite automaton constructions. *Theoret. Computer Sci.* **289** (2002), 137–163.
[10] J. H. Conway, *Regular algebra and finite machines*, Chapman and Hall, 1971.
[11] V. Glushkov, The abstract theory of automata. *Russian Mathematical Surveys* **16** (1961), 1–53.
[12] S. Lombardy and J. Sakarovitch, Derivatives of rational expressions with multiplicity, *Theoret. Computer Sci.* to appear. (Journal version of *Proc. MFCS 02*, LNCS 2420 (2002), 471–482.)
[13] R. McNaughton, and H. Yamada, Regular expressions and state graphs for automata. *IRE Trans. on electronic computers 9* (1960), 39–47.
[14] J. Sakarovitch, A construction on automata that has remained hidden, *Theoret. Computer Sci.* **204** (1998), 205–231.
[15] J. Sakarovitch, *Eléments de théorie des automates*, Vuibert, 2003. English Trans., Cambridge University Press, to appear.

[16] K. Thompson, Regular expression search algorithm, *Comm. Assoc. Comput. Mach.* **11** (1968), 419–422.

[17] D. Wood, *Theory of computation*, Wiley, 1987.

[18] S. Yu, Regular languages, in *Handbook of Formal Languages*, G. Rozenberg et A. Salomaa (éd.), vol. 1, pp. 41–111, Elsevier, 1997.