Discrete Mathematics and Theoretical Computer Science DMTCS vol. **15:2**, 2013, 95–120

Derivatives of Approximate Regular Expressions

J.-M. Champarnaud† H. Jeanne‡ L. Mignot§

LITIS, Université de Rouen, 76801 Saint-Étienne du Rouvray Cedex, France

received 26th June 2012, *revised 24th June 2013*, *accepted 24th June 2013*.

Our aim is to construct a finite automaton recognizing the set of words that are at a bounded distance from some word of a given regular language. We define new regular operators, the similarity operators, based on a generalization of the notion of distance and we introduce the family of regular expressions extended to similarity operators, that we call AREs (Approximate Regular Expressions). We set formulae to compute the Brzozowski derivatives and the Antimirov derivatives of an ARE, which allows us to give a solution to the ARE membership problem and to provide the construction of two recognizers for the language denoted by an ARE. As far as we know, the family of approximative regular expressions is introduced for the first time in this paper. Classical approximate regular expression matching algorithms are approximate matching algorithms on regular expressions. Our approach is rather to process an exact matching on approximate regular expressions.

Keywords: automata theory, regular expression derivation, similarity operators, distance operators, extension of regular expressions to similarity operators, extended regular expression derivation, membership problem, derivative-based finite automata.

1 Introduction

This paper addresses the problem of constructing a finite automaton that recognizes the language of all the words that are at a distance less than or equal to a given positive integer k from some word of a given regular language. Our approach is based on the extension of regular expressions to approximate regular expressions (AREs) that handle distance operators. More precisely, we first define a new family of operators: given an integer k, the $\mathbb{F}_k$ operator is such that, for any regular language L, the language $\mathbb{F}_k(L)$ is the set of all the words that are at a distance less than or equal to k from some word of L. We then consider the family of approximate regular expressions obtained from the family of regular expressions by adding the family of $\mathbb{F}_k$ operators to the set of regular operators. We provide a formula that, given a regular language L, computes the quotient of the language $\mathbb{F}_k(L)$ with respect to a symbol. We finally extend the computation of Brzozowski derivatives [3] (resp. of Antimirov derivatives [1]) to the family of approximate regular expressions. The first benefit of the derivation of an ARE is that it yields an elegant

†Email: `jean-marc.champarnaud@univ-rouen.fr`
‡Email: `hadrien.jeanne@univ-rouen.fr`
§Email: `ludovic.mignot@univ-rouen.fr`

solution for the approximate membership problem. Moreover, the set of Brzozowski derivatives (resp. of Antimirov derivatives) of an ARE is shown to be finite. As a consequence, the derivation of an ARE enables the computation of a finite automaton that recognizes the language of this ARE.

The similarity between two words is generally measured by a distance and two basic types of distance called Hamming distance and Levenshtein distance (or edit distance) are generally considered. In our constructions the similarity between two words is handled by a word comparison function, that is more general than a distance (for instance, a comparison function is not necessarily symmetrical). It is the reason why we will speak of similarity operators rather than of distance operators.

The aim of this paper is to investigate the properties of the AREs family, in particular to define formulae for computing the set of (Brzozowski or Antimirov) derivatives of an ARE and to check the properties of this set. This theoretical study leads to a solution for the approximate membership problem as well as to a solution for the approximate regular expression matching problem (based on the automaton associated with the set of derivatives of an ARE). However, this paper is not an algorithmic contribution to the approximate regular expression matching problem: it investigates new automaton-theoretic constructions that hopefully make a sound foundation for the design of new approximate matching algorithms, but it does not present new efficient algorithms.

Let us recall that approximate matching consists in locating the segments of the text that approximately correspond to the pattern to be matched, *i.e.* segments that do not present too many errors with respect to the pattern. This research topic has numerous applications, in biology or in linguistics for example, and many algorithms have been designed in this framework for more than thirty years especially concerning approximate string matching (see [6, 14] for a survey of such algorithms). Two contexts can be distinguished: in the off-line case, that is when a pre-computing of the text is performed, the basic tool is the construction of indexes [9]; otherwise, the basic technique is dynamic programming [12]. In both cases, automata constructions have been used, either to represent an index [18, 2] or to simulate dynamic programming [8].

Several studies address the problem of constructing a finite automaton that recognizes the language of all the words that are at a distance less than or equal to a given positive integer k from a given word. For instance this problem is considered in [7] where Hamming distance is used and in [17] where Levenshtein distance is used. A challenging problem is to tackle the more general case where the pattern is no longer a word but a regular expression [15, 19]. The solution described in [11] first computes $k+1$ clones of some non-deterministic automaton recognizing the language of the regular expression and then interconnects these clones by a set of transitions that depends on the type of distance.

As far as we know, the family of approximate regular expressions is introduced for the first time in this paper. Approximate regular expression matching algorithms described in the papers above-cited are approximate matching algorithms on regular expressions. Our approach is rather to process an exact matching on approximate regular expressions.

This paper is an extended version of [5]. Classical notions of language theory, such as derivative computation, are recalled in Section 2. Section 3 gives a formalization of the notion of word comparison function and provides a definition of the family of approximate regular expressions. The two next sections investigate derivation-based constructions of an automaton from an approximate regular expression. For seek of clarity, the standard case of Hamming and Levenshtein distances is first described and illustrated in Section 4 (without any proof), while the general case is addressed in Section 5; finally the link between the proofs of the standard case and of the general case is shown in Subsection 5.4.

2 Preliminaries

Given a set X, we denote by $\mathrm{Card}(X)$ the number of elements in X.

A *finite automaton* A is a 5-tuple $(\Sigma, Q, I, F, \delta)$ with:

- Σ the *alphabet* (a finite set of symbols),
- Q a finite set of *states*,
- $I \subset Q$ the set of *initial states*,
- $F \subset Q$ the set of *final states*,
- $\delta \subset Q \times \Sigma \times Q$ the set of *transitions*.

The set δ is equivalent to the function from $Q \times \Sigma$ to 2^Q defined by: $q' \in \delta(q,a)$ if and only if $(q,a,q') \in \delta$. The domain of the function δ is extended to $2^Q \times \Sigma^*$ as follows: $\forall P \subset Q$, $\forall a \in \Sigma$, $\forall w \in \Sigma^*$, $\delta(P,\varepsilon) = P$, $\delta(P,a) = \bigcup_{p \in P} \delta(p,a)$ and $\delta(P, a \cdot w) = \delta(\delta(P,a), w)$. The automaton A *recognizes* the language $L(A) = \{w \in \Sigma^* \mid \delta(I,w) \cap F \neq \emptyset\}$. The automaton A is *deterministic* if $\mathrm{Card}(I) = 1$ and $\forall (q,a) \in Q \times \Sigma$, $\mathrm{Card}(\delta(q,a)) \leq 1$.

A *regular expression* E over an alphabet Σ is inductively defined by:

$$E = \emptyset,\ E = \varepsilon,\ E = a,$$
$$E = (F + G),\ E = (F \cdot G),\ E = (F^*)$$

where a is any symbol in Σ and F and G are any two regular expressions.

The *language* $L(E)$ *denoted by* E is inductively defined by:

$$L(\emptyset) = \emptyset,\ L(a) = \{a\},\ L(\varepsilon) = \{\varepsilon\},$$
$$L(E+F) = L(E) \cup L(F),\ L(E \cdot F) = L(E) \cdot L(F) \text{ and } L(F^*) = (L(F))^*$$

where a is any symbol in Σ, F and G are any two regular expressions, and for any $L_1, L_2 \subset \Sigma^*$,

$$L_1 \cup L_2 = \{w \mid w \in L_1 \vee w \in L_2\},$$
$$L_1 \cdot L_2 = \{w_1 w_2 \mid w_1 \in L_1 \wedge w_2 \in L_2\}$$
$$\text{and } L_1^* = \{w_1 \cdots w_k \mid k \geq 1 \wedge \forall j \in \{1, \ldots, k\},\ w_j \in L_1\} \cup \{\varepsilon\}.$$

A language L is *regular* if there exists a regular expression E such that $L(E) = L$. It has been proved by Kleene [10] that a language is regular if and only if it is recognized by a finite automaton.

Given a language L over an alphabet Σ and a word w in Σ^*, the *membership problem* is to determine whether w belongs to L. It can be solved by the computation of the boolean $\mathrm{r}(w,L)$ defined by:

$$\mathrm{r}(w,L) = \begin{cases} 1 & \text{if } w \in L, \\ 0 & \text{otherwise.} \end{cases}$$

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

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

The quotient $w^{-1}(L)$ of L w.r.t. a word w in Σ^* is the set $\{w' \in \Sigma^* \mid w \cdot w' \in L\}$. It can be recursively computed as follows: $\varepsilon^{-1}(L) = L$, $(aw')^{-1}(L) = w'^{-1}(a^{-1}(L))$ with $a \in \Sigma$ and $w' \in \Sigma^+$. The Myhill-Nerode Theorem [13, 16] states that a language L is regular if and only if the set of quotients $\{u^{-1}(L) \mid u \in \Sigma^*\}$ is finite.

Since $\mathrm{r}(w, L) = \mathrm{r}(\varepsilon, w^{-1}(L))$, the membership problem can be solved using the quotient formulae and the following straightforward computation of $\mathrm{r}(\varepsilon, L)$:

$$\mathrm{r}(\varepsilon, \{a\}) = \mathrm{r}(\varepsilon, \emptyset) = 0, \mathrm{r}(\varepsilon, \{\varepsilon\}) = 1,$$
$$\mathrm{r}(\varepsilon, L_1 \cup L_2) = \mathrm{r}(\varepsilon, L_1) \vee \mathrm{r}(\varepsilon, L_2), \mathrm{r}(\varepsilon, L_1 \cdot L_2) = \mathrm{r}(\varepsilon, L_1) \wedge \mathrm{r}(\varepsilon, L_2),$$
$$\mathrm{r}(\varepsilon, L_1^*) = 1.$$

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

$$\frac{d}{d_a}(\emptyset) = \frac{d}{d_a}(\varepsilon) = \frac{d}{d_a}(b) = \emptyset, \quad \frac{d}{d_a}(a) = \varepsilon,$$
$$\frac{d}{d_a}(F^*) = \frac{d}{d_a}(F) \cdot F^*, \quad \frac{d}{d_a}(F + G) = \frac{d}{d_a}(F) + \frac{d}{d_a}(G)$$
$$\frac{d}{d_a}(F \cdot G) = \begin{cases} \frac{d}{d_a}(F) \cdot G + \frac{d}{d_a}(G) & \text{if } \mathrm{r}(\varepsilon, L(F)) = 1, \\ \frac{d}{d_a}(F) \cdot G & \text{otherwise.} \end{cases}$$

The derivative of E is extended to words of Σ^* as follows:

$$\frac{d}{d_\varepsilon}(E) = E, \frac{d}{d_{aw}}(E) = \frac{d}{d_w}(\frac{d}{d_a}(E)).$$

Since $w^{-1}(L(E)) = L(\frac{d}{d_w}(E))$, it holds $\mathrm{r}(w, L(E)) = \mathrm{r}(\varepsilon, L(\frac{d}{d_w}(E)))$. For convenience, we set $\mathrm{r}(w, E) = \mathrm{r}(w, L(E))$. Notice that the boolean $\mathrm{r}(\varepsilon, E)$ can be inductively computed as follows:

$$\mathrm{r}(\varepsilon, a) = \mathrm{r}(\varepsilon, \emptyset) = 0, \mathrm{r}(\varepsilon, \varepsilon) = 1,$$
$$\mathrm{r}(\varepsilon, E_1 \cup E_2) = \mathrm{r}(\varepsilon, E_1) \vee \mathrm{r}(\varepsilon, E_2), \mathrm{r}(\varepsilon, E_1 \cdot E_2) = \mathrm{r}(\varepsilon, E_1) \wedge \mathrm{r}(\varepsilon, E_2),$$
$$\mathrm{r}(\varepsilon, E_1^*) = 1.$$

As a consequence, derivation provides a syntactical solution for the membership problem.

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

$$\frac{d'}{d'_a}(\emptyset) = \frac{d'}{d'_a}(\varepsilon) = \frac{d'}{d'_a}(b) = \emptyset, \frac{d'}{d'_a}(a) = \varepsilon,$$
$$\frac{d'}{d'_a}(E + F) = (\frac{d'}{d'_a}(F) + \frac{d'}{d'_a}(G))_{\sim_s}, \frac{d'}{d'_a}(F^*) = (\frac{d'}{d'_a}(F) \cdot F^*)_{\sim_s},$$
$$\frac{d'}{d'_a}(F \cdot G) = \begin{cases} (\frac{d'}{d'_a}(F) \cdot G + \frac{d'}{d'_a}(G))_{\sim_s} & \text{if } \mathrm{r}(\varepsilon, F) = 1, \\ (\frac{d'}{d'_a}(F) \cdot G)_{\sim_s} & \text{otherwise.} \end{cases}$$

The *dissimilar derivative finite automaton* $B'(E) = (\Sigma, Q, \{q_0\}, F, \delta)$ of a regular expression E over an alphabet Σ is defined by:

- $Q = \mathcal{D}'_E$,
- $q_0 = (E)_{\sim_s}$,
- $F = \{q \in Q \mid \varepsilon \in L(q)\}$,
- $\delta = \{(q, a, q') \in Q \times \Sigma \times Q \mid \frac{d'}{d'_a}(q) = q'\}$.

The automaton $B'(E)$ is deterministic and it recognizes the language $L(E)$. Its size can be exponentially larger than the number of symbols of E.

Antimirov's algorithm [1] constructs a finite automaton from a regular expression E. It is based on the *partial derivative* computation. The partial derivative of a regular expression E w.r.t. a symbol a is the set $\frac{\partial}{\partial_a}(E)$ of expressions defined as follows:

$$\frac{\partial}{\partial_a}(\emptyset) = \frac{\partial}{\partial_a}(\varepsilon) = \frac{\partial}{\partial_a}(b) = \emptyset,\ \frac{\partial}{\partial_a}(a) = \{\varepsilon\},$$
$$\frac{\partial}{\partial_a}(F+G) = \frac{\partial}{\partial_a}(F) \cup \frac{\partial}{\partial_a}(G),\ \frac{\partial}{\partial_a}(F^*) = \frac{\partial}{\partial_a}(F)\cdot F^*,$$
$$\frac{\partial}{\partial_a}(F\cdot G) = \begin{cases} \frac{\partial}{\partial_a}(F)\cdot G \cup \frac{\partial}{\partial_a}(G) & \text{if } \mathrm{r}(\varepsilon, F) = 1, \\ \frac{\partial}{\partial_a}(F)\cdot G & \text{otherwise,} \end{cases}$$

with for any set $\mathcal{E}$ of expressions, $\mathcal{E}\cdot F = \bigcup_{E\in\mathcal{E}} E\cdot F$.
The partial derivative of E is extended to words of Σ^* as follows:

$$\frac{\partial}{\partial_\varepsilon}(E) = \{E\},\ \frac{\partial}{\partial_{aw}}(E) = \frac{\partial}{\partial_w}\left(\frac{\partial}{\partial_a}(E)\right),$$

with for a set $\mathcal{E}$ of expressions, $\frac{\partial}{\partial_a}(\mathcal{E}) = \bigcup_{E\in\mathcal{E}} \frac{\partial}{\partial_a}(E)$. Every element of the partial derivative of E w.r.t. a word w in Σ^* is called a *derivated term of* E w.r.t. w. The *set of the derivated terms of* E is the union of the sets of the derivated terms of E w.r.t. w, for all w in Σ^*. Antimirov [1] has shown that the size $\mathrm{Card}(\mathcal{DT}_E)$ of the set $\mathcal{DT}_E$ of the derivated terms of E is at most $n+1$, where n is the number of symbols of E.

Furthermore, for any word w in Σ^*, $\bigcup_{E'\in\frac{\partial}{\partial_w}(E)} L(E') = w^{-1}(L(E))$. Consequently, the partial derivation provides another syntactical solution for the membership problem as well as a finite automaton computation. Indeed, it can be shown that $\mathrm{r}(w,E) = \bigvee_{E'\in\frac{\partial}{\partial_w}(E)} \mathrm{r}(\varepsilon, E')$.

The *derivated term finite automaton* $A(E) = (\Sigma, Q, \{q_0\}, F, \delta)$ of a regular expression E is defined as follows:

- $Q = \mathcal{DT}_E$,
- $q_0 = E$,
- $F = \{q\in Q \mid \mathrm{r}(\varepsilon, q) = 1\}$,
- $\delta = \{(q,a,q')\in Q\times\Sigma\times Q \mid q'\in\frac{\partial}{\partial_a}(q)\}$.

The automaton $A(E)$ recognizes the language $L(E)$.

In this paper, we consider the *approximate membership problem* that is defined as follows:

Given a regular expression E over an alphabet Σ, a word w in Σ^*, a function $\mathbb{F}$ from $\Sigma^*\times\Sigma^*$ to $\mathbb{N}$ and an integer k, is there a word w' in $L(E)$ satisfying $\mathbb{F}(w,w')\leq k$?

In the following, we provide a syntactical solution for the approximate membership problem in the case where the function $\mathbb{F}$ satisfies specific properties.

3 Comparison Functions: Symbols, Sequences and Words

Let Σ be an alphabet, $S = \Sigma\cup\{\varepsilon\}$ and X be a subset of $S\times S$. A *cost function* C over X is a function from X to $\mathbb{N}$ satisfying **Condition 1:** for all α in S, $\mathrm{C}(\alpha,\alpha) = 0$. For any pair (α,β) in $S\times S$ such that $\mathrm{C}(\alpha,\beta)$ is not defined, let us set $\mathrm{C}(\alpha,\beta) = \perp$. Consequently, a cost function can be viewed as a function from $S\times S$ to $\mathbb{N}\cup\{\perp\}$ satisfying Condition 1. Since we use $\perp$ to deal with undefined computation, we set for all x in $\mathbb{N} \cup \{\perp\}$, $\perp + x = x + \perp = x - \perp = \perp - x = \perp$ and for all integers x, y in $\mathbb{N}$, $x - y = \perp$ when $y > x$. A cost function can be represented by a directed and labelled graph $\mathrm{C} = \{S, V\}$ where V is a subset of $S \times (\mathbb{N} \cup \{\perp\}) \times S$ such that for all (α, β) in $S \times S$, $\mathrm{C}(\alpha, \beta) = k \Leftrightarrow (\alpha, k, \beta) \in V$. Transitions labelled by $\perp$ can be omitted in the graphical representation, as well as the implicit transitions $(\alpha, 0, \alpha)$ (See Example 1).

Example 1 *Let* $\Sigma = \{a, b, c\}$. *Let* C *be the cost function defined as follows:*

$$\mathrm{C}(x,y) = \begin{cases} 0 & \textit{if } x = y, \\ 4 & \textit{if } x = a \wedge y = c, \\ 3 & \textit{if } x = c \wedge y = a, \\ 1 & \textit{if } x \in \{a, c\} \wedge y = b, \\ \perp & \textit{otherwise.} \end{cases}$$

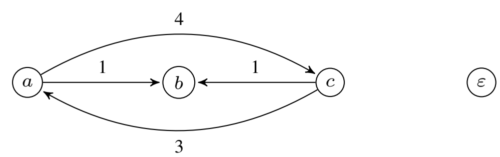

Fig. 1: The cost function C

The cost function C *can be represented by the graph in Figure 1.*

Given a positive integer k we now consider the set S^k of all the sequences $s = (s_1, \ldots, s_k)$ of size k made of elements of S. A *sequence comparison function* is a function $\mathcal{F}$ from $\bigcup_{k>0} S^k \times S^k$ to $\mathbb{N} \cup \{\perp\}$. Given a pair (s, s') of sequences with the same size, $\mathcal{F}(s, s')$ either is an integer or is undefined. In the following we will consider sequence comparison functions $\mathcal{F}$ satisfying **Condition 2:** for any couple (α, β) in $S \times S$, $\mathcal{F}(\alpha, \beta) = \mathrm{C}(\alpha, \beta)$, where C is some cost function C over $S \times S$, and **Condition 3:** $\mathcal{F}$ is a *symbol-wise* comparison function, that is, for any two sequences $s = (s_1, \ldots, s_n)$ and $s' = (s'_1, \ldots, s'_n)$, it holds:

$$\mathcal{F}(s, s') = \mathcal{F}((s_1), (s'_1)) + \mathcal{F}((s_2, \ldots, s_n), (s'_2, \ldots, s'_n)) = \sum_{k \in \{1,\ldots,n\}} \mathcal{F}((s_k), (s'_k)).$$

We consider that those functions satisfy Condition 1 , *i.e.* for all α in S, $\mathcal{F}((\alpha), (\alpha)) = 0$. Consequently, for any pair of sequences $s = (s_1, \ldots, s_k)$ and $s' = (s'_1, \ldots, s'_k)$ such that $k > 1$, **Condition 4** is satisfied: if there exists an integer k' in $\{1, \ldots, k\}$ such that $s_{k'} = s'_{k'}$, then:

$$\mathcal{F}(s, s') = \begin{cases} \mathcal{F}((s_2, \ldots, s_k), (s'_2, \ldots, s'_k)) & \text{if } k' = 1, \\ \mathcal{F}((s_1, \ldots, s_{k-1}), (s'_1, \ldots, s'_{k-1})) & \text{if } k' = k, \\ \mathcal{F}((s_1, \ldots, s_{k'-1}, s_{k'+1}, \ldots, s_k), (s'_1, \ldots, s'_{k'-1}, s'_{k'+1}, \ldots, s'_k)) & \text{otherwise.} \end{cases}$$

As a consequence of Condition 3, a symbol-wise sequence comparison function is defined by the images of the pairs of sequences of size 1. Notice that a sequence comparison function is not necessarily symbol-wise, *e.g.* for a given cost function F, $\mathcal{F}((s_1, \ldots, s_n), (s'_1, \ldots, s'_n)) = \sum_{k \in \{1,\ldots,n\}} \mathrm{F}(s_k, s'_k)^k$.

Two of the most well-known symbol-wise sequence comparison functions are the Hamming one ($\mathcal{H}$) and the Levenshtein one ($\mathcal{L}$) respectively defined for any integer $n > 0$ and for any pair of sequences $s = (s_1, \ldots, s_n)$ and $s' = (s'_1, \ldots, s'_n)$ in $S^n \times S^n$ by:

$$\mathcal{H}(s, s') = \sum_{k \in \{1,\ldots,n\}} \mathrm{H}(s_k, s'_k), \quad \mathcal{L}(s, s') = \sum_{k \in \{1,\ldots,n\}} \mathrm{L}(s_k, s'_k),$$

with H and L the two cost functions respectively defined for all a, b in $\Sigma \cup \{\varepsilon\}$ by:

$$\mathrm{H}(a,b) = \begin{cases} \perp & \text{if } (a = \varepsilon \vee b = \varepsilon) \wedge (a, b) \neq (\varepsilon, \varepsilon), \\ 1 & \text{if } a \neq b, \\ 0 & \text{otherwise,} \end{cases} \quad \text{and } \mathrm{L}(a,b) = \begin{cases} 1 & \text{if } a \neq b, \\ 0 & \text{otherwise.} \end{cases}$$

Let us now explain how a word comparison function can be deduced from a sequence comparison function. Let w be a word in Σ^* and $|w|$ be its *length*. The sequence $s = (s_1, \ldots, s_n)$ in S^n is said to be a *split-up* of w if $s_1 \cdots s_n = w$. The integer n is the *size* of s. The set of all the split-ups of size k of a word w is denoted by $\mathrm{Split}_k(w)$ and the set of all the split-ups of w is denoted by $\mathrm{Split}(w)$.

Let $\mathcal{F}$ be a sequence comparison function, (u, v) be a pair of words of Σ^*, and k be a positive integer. We consider the following sets:

$$Y(u,v) = \{\mathcal{F}(u',v') \mid \exists k \in \mathbb{N}, k \geq 1 \wedge (u',v') \in \mathrm{Split}_k(u) \times \mathrm{Split}_k(v)\} \cap \mathbb{N},$$
$$Y_m(u,v) = \{\mathcal{F}(u',v') \mid \exists k \in \mathbb{N}, 1 \leq k \leq m \wedge (u',v') \in \mathrm{Split}_k(u) \times \mathrm{Split}_k(v)\} \cap \mathbb{N}.$$

Definition 1 *Let $\mathcal{F}$ be a sequence comparison function. The* word comparison function *associated with $\mathcal{F}$ is the function $\mathbb{F}$ from $\Sigma^* \times \Sigma^*$ to $\mathbb{N} \cup \{\perp\}$ defined by:*

$$\mathbb{F}(u,v) = \min\{Y(u,v)\} \text{ if } Y(u,v) \neq \emptyset, \qquad \mathbb{F}(u,v) = \perp \text{ otherwise.}$$

Notice that a word comparison function is not necessarily symmetrical. Indeed, some problems can be modelized with a non-symmetrical function. For instance, given two words w and w', can w be obtained from w' by deleting some letters, *i.e.* is w a subword of w'? Such a problem can be modelized by the word comparison function $\mathbb{D}$ associated to the symbol-wise comparison function $\mathcal{D}$ defined for any pair of sequences of length 1 by:

$$\forall(\alpha,\beta) \in (\Sigma \cup \{\varepsilon\})^2, \mathcal{D}((\alpha),(\beta)) = \begin{cases} 0 & \text{if } \alpha = \beta, \\ 1 & \text{if } \alpha = \varepsilon \wedge \beta \in \Sigma, \\ \perp & \text{otherwise.} \end{cases}$$

It can be shown that for any two words w and w' in Σ^*:

$$\mathbb{D}(w,w') = \begin{cases} \perp & \text{if } w \text{ is not a subword of } w', \\ |w'| - |w| & \text{otherwise.} \end{cases}$$

In the case of a sequence comparison function based on a cost function, the whole set $\mathbb{N}$ needs not to be considered. Indeed, according to Condition 4, if $u \neq \varepsilon$ or $v \neq \varepsilon$, then $Y(u,v) = Y_{|u|+|v|}(u,v)$ and we can write:

$$\mathbb{F}(u,v) = \begin{cases} 0 & \text{if } u = v = \varepsilon, \\ \min\{Y_{|u|+|v|}(u,v)\} & \text{if } (u,v) \neq (\varepsilon,\varepsilon) \wedge Y_{|u|+|v|}(u,v) \neq \emptyset, \\ \perp & \text{otherwise.} \end{cases}$$

The *Hamming distance* $\mathbb{H}$ and the *Levenshtein distance* $\mathbb{L}$ are the word comparison functions respectively associated to the sequence comparison functions $\mathcal{H}$ and $\mathcal{L}$. Both of them satisfy the properties of word distances[(i)]. Notice that in the following we will handle word comparison functions that are not necessarily distances (see Example 1 for the definition of a nonsymmetrical cost function).

Example 2 *Let C be the cost function defined in Example 1. Let $s = (s_1)$ and $s' = (s'_1)$ be two sequences of size 1. We define four symbol-wise sequence comparison functions by setting the images of the pairs of sequences of size 1 from the cost function* C.

$$\rightarrow^{\mathrm{C}} (s,s') = \mathrm{C}(s_1,s'_1), \quad \leftrightarrow^{\mathrm{C}} (s,s') = \min\{\mathrm{C}(s_1,s'_1), \mathrm{C}(s'_1,s_1)\},$$
$$\leftarrow^{\mathrm{C}} (s,s') = \mathrm{C}(s'_1,s_1), \quad \rightrightarrows^{\mathrm{C}} (s,s') = \min_{x \in \Sigma \cup \{\varepsilon\}}\{\mathrm{C}(s_1,x) + \mathrm{C}(s'_1,x)\}.$$

Let us consider the two split-ups $t = (a,c,a)$ and $t' = (c,a,c)$. According to Figure 2, it holds:

(i) A *word distance* $\mathbb{D}$ is a word comparison function satisfying the three following properties for all $x, y, z \in \Sigma^*$: **(1)** $\mathbb{D}(x,y) = 0 \Rightarrow x = y$, **(2)** $\mathbb{D}(x,y) = \mathbb{D}(y,x)$, **(3)** $\mathbb{D}(x,y) + \mathbb{D}(y,z) \geq \mathbb{D}(x,z)$.

$$\begin{aligned}&\rightarrow^{\mathrm{C}}(t,t')=11,\\&\leftarrow^{\mathrm{C}}(t,t')=10,\\&\leftrightarrow^{\mathrm{C}}(t,t')=9,\\&\rightrightarrows^{\mathrm{C}}(t,t')=6.\end{aligned}$$

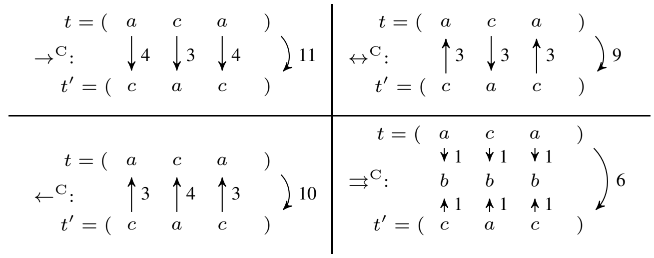

Fig. 2: Examples of sequence comparisons

Any word comparison function can be used as a language operator in order to compute the set of words that are at a bounded distance from some word of a given language.

Definition 2 *Let L be a language over an alphabet Σ, $\mathbb{F}$ a word comparison function and k an element in $\mathbb{N}\cup\{\perp\}$. Then:*

$$\mathbb{F}_k(L)=\begin{cases}\{w\in\Sigma^*\mid \exists u\in L,\mathbb{F}(w,u)\in\{0,\ldots,k\}\} & \textit{if } k\in\mathbb{N},\\ \emptyset & \textit{otherwise.}\end{cases}$$

The operator $\mathbb{F}_k$ is called a *similarity operator*. Let us notice that $\mathbb{F}_k(\mathbb{F}_{k'}(L))$ is not necessarily equal to $\mathbb{F}_{k+k'}(L)$. Indeed, let us consider the three languages $L_1=\mathbb{F}_1(\{a\})$, $L_2=\mathbb{F}_1(\mathbb{F}_1(\{a\}))$ and $L_3=\mathbb{F}_2(\{a\})$ over the alphabet $\Sigma=\{a,b\}$ with $\mathbb{F}$ the word comparison function associated with the symbol-wise sequence comparison function $\mathcal{F}$ defined for any symbol α,β by $\mathcal{F}((\alpha),(\beta))=0$ if $\alpha=\beta$, $\mathcal{F}((\alpha),(\beta))=2$ otherwise. Then $L_1=L_2=\{a\}$ whereas $L_3=\{\varepsilon,a,b,aa,ab,ba\}$.

Definition 3 *An* approximate regular expression[(ii)] ***(ARE)*** *E over an alphabet Σ is inductively defined by:*

$$E=\emptyset,\ E=\varepsilon,\ E=a,$$
$$E=F+G,\ E=(F\cdot G),\ E=(F^*),$$
$$E=\mathbb{F}_k(F)$$

where a is any symbol in Σ, F and G are any two AREs, $\mathbb{F}$ is any symbol-wise word comparison function and k is any element in $\mathbb{N}\cup\{\perp\}$.

Definition 4 *The* language denoted *by an ARE E is the language $L(E)$ inductively defined by:*

$$L(\emptyset)=\emptyset,\ L(\varepsilon)=\{\varepsilon\},\ L(a)=\{a\},$$
$$L(F+G)=L(F)\cup L(G),\ L(F\cdot G)=L(F)\cdot L(G),\ L(F^*)=L(F)^*,$$
$$L(\mathbb{F}_k(F))=\mathbb{F}_k(L(F)),$$

where a is any symbol in Σ, F and G are any two AREs, $\mathbb{F}$ is any symbol-wise word comparison function and k is any element in $\mathbb{N}\cup\{\perp\}$.

In order to prove that the language denoted by an ARE E is regular, we will show how to compute a finite automaton recognizing $L(E)$.

(ii) The fact that any ARE denotes a regular language is proved in Corollary 1.

4 Hamming and Levenshtein Derivation Formulae

In this section, we extend the derivation formulae to the family of approximate regular expressions where the word comparison functions are the usual Hamming and Levenshtein distances. Notice that the proofs are not given in this section, but will be stated in Section 5.4, deduced from the proof of the general case provided in Section 5.

Let a be a symbol in an alphabet Σ and L be a regular language over Σ. Let k be an integer and $L' = \mathbb{L}_k(L)$. The quotient of L' w.r.t. a is by definition the set of words w such that there exists a word w' in L satisfying $\mathbb{L}(aw, w') \leq k$. Consequently, we distinguish the four following cases, according to the way w' can be split:

1. $w' = aw''$ and $\mathbb{L}(a, a) + \mathbb{L}(w, w'') \leq k$: hence the word w'' is by definition in $a^{-1}(L)$ and $\mathbb{L}(w, w'') \leq k$. Consequently, $w \in \mathbb{L}_k(a^{-1}(L))$;
2. $w' = bw''$ with $b \in \Sigma \setminus \{a\}$ and $\mathbb{L}(a, b) + \mathbb{L}(w, w'') \leq k$: hence the word w'' is by definition in $b^{-1}(L)$ and $\mathbb{L}(w, w'') \leq k - 1$. Consequently, $w \in \mathbb{L}_{k-1}(b^{-1}(L))$;
3. $\mathbb{L}(a, \varepsilon) + \mathbb{L}(w, w') \leq k$: hence the word w' is by definition in L and $\mathbb{L}(w, w') \leq k - 1$. Consequently, $w \in \mathbb{L}_{k-1}(L)$;
4. $w' = bw''$ with $b \in \Sigma$ and $\mathbb{L}(\varepsilon, b) + \mathbb{L}(aw, w'') \leq k$: hence the word w'' is by definition in $b^{-1}(L)$ and $\mathbb{L}(aw, w'') \leq k - 1$. Consequently, $w \in a^{-1}(\mathbb{L}_{k-1}(b^{-1}(L)))$.

Notice that for the Hamming distance, only the two first cases need to be considered since $\mathbb{H}(\alpha, \beta) = \perp$ whenever $\alpha = \varepsilon$ and $\beta \neq \varepsilon$ or $\alpha \neq \varepsilon$ and $\beta = \varepsilon$.

As a consequence, the following lemma can be stated.

Lemma 1 *Let L be a regular language over an alphabet Σ, a be a symbol in Σ and k be an element in $\mathbb{N} \cup \{\perp\}$. Then:*

$$a^{-1}(\mathbb{H}_k(L)) = \mathbb{H}_k(a^{-1}(L)) \cup \bigcup_{b \in \Sigma \setminus \{a\}} \mathbb{H}_{k-1}(b^{-1}(L)),$$

$$a^{-1}(\mathbb{L}_k(L)) = \left(\begin{array}{l} \quad \mathbb{L}_k(a^{-1}(L)) \\ \cup \bigcup_{b \in \Sigma \setminus \{a\}} \mathbb{L}_{k-1}(b^{-1}(L)) \\ \cup\, \mathbb{L}_{k-1}(L) \\ \cup\, a^{-1}(\bigcup_{b \in \Sigma} \mathbb{L}_{k-1}(b^{-1}(L))) \end{array} \right).$$

In the remaining of this section, we consider restricted AREs that only use Hamming and Levenshtein distances.

Definition 5 *Let Σ be an alphabet. A* Hamming-Levenshtein Approximate Regular Expression *(HLARE) over Σ is an ARE E over Σ satisfying the following condition:*

For any subexpression G of E such that $G = \mathbb{F}_k(H)$, either $\mathbb{F} = \mathbb{H}$ or $\mathbb{F} = \mathbb{L}$.

4.1 Brzozowski Derivatives for an HLARE

In this subsection, we extend the Brzozowski derivation to the HLAREs. From an HLARE E and a word w, Brzozowski derivation allows us to syntactically compute an HLARE $D'_w(E)$, called the dissimilar derivative of E w.r.t. w, denoting the language $w^{-1}(L(E))$.

Definition 6 *Let E be an HLARE over an alphabet Σ. Let a and b be two distinct symbols in Σ and w be a word in Σ^*. The* dissimilar derivative *of E w.r.t. the symbol a (resp. the word w) is the HLARE $D'_a(E)$ (resp. $D'_w(E)$) defined as follows:*

$$D'_a(\varepsilon) = D'_a(\emptyset) = D'_a(b) = \emptyset,\ D'_a(a) = \varepsilon,$$
$$D'_a(E_1 + E_2) = (D'_a(E_1) + D'_a(E_2))_{\sim_s},\ D'_a(E_1^*) = (D'_a(E_1) \cdot E_1^*)_{\sim_s},$$
$$D'_a(E_1 \cdot E_2) = \begin{cases} (D'_a(E_1) \cdot E_2 + D'_a(E_2))_{\sim_s} & \text{if } \mathrm{r}(\varepsilon, E_1) = 1, \\ (D'_a(E_1) \cdot E_2)_{\sim_s} & \text{if } \mathrm{r}(\varepsilon, E_1) = 0, \end{cases}$$
$$D'_a(\mathbb{H}_k(E_1)) = (\mathbb{H}_k(D'_a(E_1)) + \textstyle\sum_{b \in \Sigma \setminus \{a\}} \mathbb{H}_{k-1}(D'_b(E_1)))_{\sim_s},$$
$$D'_a(\mathbb{L}_k(E_1)) = \begin{pmatrix} \mathbb{L}_k(D'_a(E_1)) \\ + \sum_{b \in \Sigma \setminus \{a\}} \mathbb{L}_{k-1}(D'_b(E_1)) \\ + \mathbb{L}_{k-1}(E_1) \\ + D'_a(\sum_{b \in \Sigma} \mathbb{L}_{k-1}(D'_b(E_1))) \end{pmatrix}_{\sim_s},$$
$$D'_w(E) = \begin{cases} E & \text{if } w = \varepsilon, \\ D'_u(D'_a(E)) & \text{if } w = au \wedge a \in \Sigma \ \wedge\ u \in \Sigma^*, \end{cases}$$

where E_1 and E_2 are any two HLAREs and k is any element in $\mathbb{N} \cup \{\perp\}$.

Lemma 2 *Let E be an HLARE over an alphabet Σ. Let w be a word in Σ^*. Then:*

$$L(D'_w(E)) = w^{-1}(L(E)).$$

The next lemma shows that the boolean $\mathrm{r}(\varepsilon, E)$ is syntactically computable for any HLARE E using dissimilar derivatives.

Lemma 3 *Let $E = \mathbb{H}_k(E')$ and $F = \mathbb{L}_k(F')$ be two HLAREs over an alphabet Σ. Then the two following propositions are satisfied:*

- $\varepsilon \in L(E) \Leftrightarrow \varepsilon \in L(E')$,
- $\varepsilon \in L(F) \Leftrightarrow \varepsilon \in L(F') \cup \bigcup_{a \in \Sigma} L(\mathbb{L}_{k-1}(D'_a(F')))$.

Given an HLARE E, we denote by $\mathcal{D}_{HL}(E)$ the set $\{D'_w(E) \mid w \in \Sigma^*\}$ of the dissimilar derivatives of E.

Lemma 4 *The set $\mathcal{D}_{HL}(E)$ of dissimilar derivatives of an HLARE E is finite.*

From this finite set, one can compute a deterministic finite automaton that recognizes $L(E)$.

Definition 7 *Let E be an HLARE over an alphabet Σ. The tuple $B'(E) = (\Sigma, Q, I, F, \delta)$ is defined by:*

- $Q = \mathcal{D}_{HL}(E)$,
- $I = \{(E)_{\sim_s}\}$,
- $F = \{q \in Q \mid \mathrm{r}(\varepsilon, q) = 1\}$,
- $\forall (q, a) \in Q \times \Sigma$, $\delta(q, a) = \{D'_a(q)\}$.

Proposition 1 *Let E be an HLARE over an alphabet Σ. Then:*

$B'(E)$ is a deterministic finite automaton that recognizes $L(E)$.

For any HLARE E, the automaton $B'(E)$ is called the *dissimilar derivative finite automaton* of E.

Example 3 presents the computation of the dissimilar derivative automaton of an HLARE. Example 4 illustrates the computation of the boolean $\mathrm{r}(w, E)$ for an HLARE E. Notice that in both examples, the following reductions are used:

$$E + \emptyset = \emptyset + E = E,$$
$$E \cdot \emptyset = \emptyset \cdot E = \emptyset,$$
$$E \cdot \varepsilon = \varepsilon \cdot E = E,$$
$$\mathbb{F}_{\perp}(E) = \emptyset.$$

Example 3 *Let $F = b^*(a+b)c^*$ and $E = \mathbb{H}_1(F)$ be an HLARE over $\Sigma = \{a, b, c\}$. The dissimilar derivatives of E are the following expressions:*

$D'_a(E) = \mathbb{H}_0(F) + \mathbb{H}_1(c^*) + \mathbb{H}_0(c^*) = E_1$	$D'_a(E_1) = \mathbb{H}_0(c^*) = E_4$
$D'_b(E) = E + \mathbb{H}_1(c^*) + \mathbb{H}_0(c^*) = E_2$	$D'_b(E_1) = \mathbb{H}_0(F) + \mathbb{H}_0(c^*) = E_3$
$D'_c(E) = \mathbb{H}_0(F) + \mathbb{H}_0(c^*) = E_3$	$D'_c(E_1) = \mathbb{H}_1(c^*) + \mathbb{H}_0(c^*) = E_5$
$D'_a(E_2) = \mathbb{H}_0(F) + \mathbb{H}_1(c^*) + \mathbb{H}_0(c^*) = E_1$	$D'_a(E_3) = \mathbb{H}_0(c^*) = E_4$
$D'_b(E_2) = E + \mathbb{H}_1(c^*) + \mathbb{H}_0(c^*) = E_2$	$D'_b(E_3) = \mathbb{H}_0(F) + \mathbb{H}_0(c^*) = E_3$
$D'_c(E_2) = \mathbb{H}_0(F) + \mathbb{H}_0(c^*) + \mathbb{H}_1(c^*) = E_1$	$D'_c(E_3) = \mathbb{H}_0(c^*) = E_4$
$D'_a(E_4) = \emptyset$	$D'_a(E_5) = \mathbb{H}_0(c^*) = E_4$
$D'_b(E_4) = \emptyset$	$D'_b(E_5) = \mathbb{H}_0(c^*) = E_4$
$D'_c(E_4) = \mathbb{H}_0(c^*) = E_4$	$D'_c(E_5) = \mathbb{H}_1(c^*) + \mathbb{H}_0(c^*) = E_5$

The dissimilar derivative automaton of E is given Figure 3.

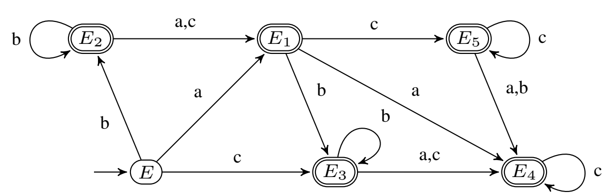

Fig. 3: The dissimilar derivative automaton of $E = \mathbb{H}_1(b^*(a+b)c^*)$

Example 4 *Let $G = \mathbb{L}_1((aba + abb)a(a)^*)$ be an HLARE over the alphabet $\Sigma = \{a, b\}$ and $w = aba$ be a word in Σ^*. Then: $\mathrm{r}(aba, G) = \mathrm{r}(\varepsilon, D'_{aba}(G))$. Let us first compute the HLARE $D'_{aba}(G)$:*

$D'_a(G) = \mathbb{L}_1((ba + bb)a(a)^*) + \mathbb{L}_0((aba + abb)a(a)^*) = G_1$

$D'_b(G_1) = \mathbb{L}_1((a + b)a(a)^*) + \mathbb{L}_0((ba + bb)a(a)^*) + \mathbb{L}_0(a(a)^*) = G_2$

$D'_a(G_2) = \mathbb{L}_1(a(a)^*) + \mathbb{L}_0(a(a)^*) + \mathbb{L}_0((a + b)a(a)^*) + \mathbb{L}_0((a)^*) = G_3$

Hence $\mathrm{r}(aba, G) = \mathrm{r}(\varepsilon, G_3)$. Furthermore, since $\varepsilon \in L(\mathbb{L}_0(D'_a(a(a)^*)))$, it holds that $\varepsilon \in L(\mathbb{L}_1(a(a)^*))$. Consequently, $\mathrm{r}(\varepsilon, G_3) = 1$ and aba belongs to $L(G)$.

Notice that in this case:

1. *The word w is split up into $s_w = (a, b, a, \varepsilon)$;*
2. *The word $w' = abaa$ in $L((aba + abb)a(a)^*)$ can be split up into $s_{w'} = (a, b, a, a)$;*

3. It holds $\mathcal{L}(s_w, s_{w'}) = 1$.

Another split-up is presented in Example 6.

4.2 Antimirov Partial Derivatives of an HLARE

In this subsection, we extend the Antimirov derivation to the HLAREs. From an HLARE E and a word w, Antimirov derivation allows us to compute a set $\Delta_w(E)$ of HLAREs, called the partial derivative of E w.r.t. w. Any HLARE in $\Delta_w(E)$ is called a derivated term of E w.r.t. w. Finally, we state that the union of the languages denoted by the derivated terms in $\Delta_w(E)$ is equal to $w^{-1}(L(E))$.

Definition 8 *Let E be an HLARE over an alphabet Σ. Let a and b be two distinct symbols in Σ and w be a word in Σ^*. The* partial derivative *of E w.r.t. the symbol a (resp. to the word w) is the set $\Delta_a(E)$ (resp. $\Delta_w(E)$) of HLAREs defined as follows:*

$$\Delta_a(\varepsilon) = \Delta_a(\emptyset) = \Delta_a(b) = \emptyset,\ \Delta_a(a) = \{\varepsilon\},$$
$$\Delta_a(E_1 + E_2) = \Delta_a(E_1) \cup \Delta_a(E_2),\ \Delta_a(E_1^*) = \Delta_a(E_1) \cdot E_1^*,$$
$$\Delta_a(E_1 \cdot E_2) = \begin{cases} \Delta_a(E_1) \cdot E_2 \cup \Delta_a(E_2) & \text{if } \mathrm{r}(\varepsilon, E_1) = 1, \\ \Delta_a(E_1) \cdot E_2 & \text{if } \mathrm{r}(\varepsilon, E_1) = 0, \end{cases}$$
$$\Delta_a(\mathbb{H}_k(E_1)) = \mathbb{H}_k(\Delta_a(E_1)) \cup \bigcup_{b\in\Sigma\setminus\{a\}} \mathbb{H}_{k-1}(\Delta_b(E_1)),$$
$$\Delta_a(\mathbb{L}_k(E_1)) = \begin{pmatrix} \mathbb{L}_k(\Delta_a(E_1)) \\ \cup \bigcup_{b\in\Sigma\setminus\{a\}} \mathbb{L}_{k-1}(\Delta_b(E_1)) \\ \cup \{\mathbb{L}_{k-1}(E_1)\} \\ \cup\, \Delta_a(\bigcup_{b\in\Sigma} \mathbb{L}_{k-1}(\Delta_b(E_1)) \end{pmatrix},$$
$$\Delta_w(E) = \begin{cases} \{E\} & \text{if } w = \varepsilon, \\ \Delta_{w'}(\Delta_a(E)) & \text{if } w = aw' \wedge a \in \Sigma\ \wedge\ w' \in \Sigma^*, \end{cases}$$

where E_1 and E_2 are any two HLAREs and k an element in $\mathbb{N} \cup \{\perp\}$ and where for any set $\mathcal{E}$ of HLAREs, for any HLARE F, for any symbol a in Σ,

$$\mathcal{E} \cdot F = \bigcup_{E\in\mathcal{E}} \{E \cdot F\},$$
$$\Delta_a(\mathcal{E}) = \bigcup_{E\in\mathcal{E}} \Delta_a(E),$$
$$\mathbb{H}_k(\mathcal{E}) = \bigcup_{E\in\mathcal{E}} \{\mathbb{H}_k(E)\},$$
$$\mathbb{L}_k(\mathcal{E}) = \bigcup_{E\in\mathcal{E}} \{\mathbb{L}_k(E)\}.$$

Lemma 5 *Let E be an HLARE over an alphabet Σ. Let w be a word in Σ^*. Then:*

$$\bigcup_{G\in\Delta_w(E)} L(G) = w^{-1}(L(E)).$$

Next lemma shows that the boolean $\mathrm{r}(\varepsilon, E)$ is syntactically computable for any HLARE E using partial derivation.

Lemma 6 *Let $E = \mathbb{H}_k(E')$ and $F = \mathbb{L}_k(F')$ be two HLAREs over an alphabet Σ. Then the two following conditions are satisfied:*

- $\varepsilon \in L(E) \Leftrightarrow \varepsilon \in L(E')$,
- $\varepsilon \in L(F) \Leftrightarrow \varepsilon \in L(F') \cup \bigcup_{a\in\Sigma, G\in\Delta_a(F')} L(\mathbb{L}_{k-1}(G))$.

Given an HLARE E, we denote by $\mathcal{DT}_{HL}(E)$ the set $\bigcup_{w\in\Sigma^*} \Delta_w(E)$ of the derivated terms of E.

Lemma 7 *The set $\mathcal{DT}_{HL}(E)$ of the derivated terms of an HLARE E is finite.*

From this finite set, one can compute a finite automaton that recognizes $L(E)$.

Definition 9 *Let E be an HLARE over an alphabet Σ. The tuple $A(E) = (\Sigma, Q, I, F, \delta)$ is defined by:*

- $Q = \mathcal{DT}_{HL}(E)$,
- $I = \{E\}$,
- $F = \{q \in Q \mid \mathrm{r}(\varepsilon, q) = 1\}$,
- $\forall (q, a) \in Q \times \Sigma$, $\delta(q, a) = \Delta_a(q)$.

Proposition 2 *Let E be an HLARE over an alphabet Σ. Then:*
$A(E)$ is a finite automaton that recognizes $L(E)$.

For any HLARE E, the automaton $A(E)$ is the *derivated term finite automaton* of E.

Example 5 presents the computation of the derivated term automaton of an HLARE. Example 6 illustrates the computation of the boolean $\mathrm{r}(w, E)$ for an HLARE E. Notice that in both of these examples, the five following reductions are used:

$$\begin{array}{c} E + \emptyset = \emptyset + E = E, \\ E \cdot \emptyset = \emptyset \cdot E = \emptyset, \\ E \cdot \varepsilon = \varepsilon \cdot E = E, \\ \mathbb{F}_{\perp}(E) = \emptyset, \\ \{\emptyset\} = \emptyset.^{\text{(iii)}} \end{array}$$

Example 5 *Let E be the HLARE defined in Example 3. The partial derivatives of E are the following sets of expressions:*

$$\begin{array}{r|l} \Delta_a(E) = \{\mathbb{H}_0(F), \mathbb{H}_1(c^*), \mathbb{H}_0(c^*)\} & \Delta_a(\mathbb{H}_1(c^*)) = \{\mathbb{H}_0(c^*)\} \\ \Delta_b(E) = \{E, \mathbb{H}_1(c^*), \mathbb{H}_0(c^*)\} & \Delta_b(\mathbb{H}_1(c^*)) = \{\mathbb{H}_0(c^*)\} \\ \Delta_c(E) = \{\mathbb{H}_0(F), \mathbb{H}_0(c^*)\} & \Delta_c(\mathbb{H}_1(c^*)) = \{\mathbb{H}_1(c^*)\} \\ \Delta_a(\mathbb{H}_0(F)) = \{\mathbb{H}_0(c^*)\} & \Delta_a(\mathbb{H}_0(c^*)) = \emptyset \\ \Delta_b(\mathbb{H}_0(F)) = \{\mathbb{H}_0(F), \mathbb{H}_0(c^*)\} & \Delta_b(\mathbb{H}_0(c^*)) = \emptyset \\ \Delta_c(\mathbb{H}_0(F)) = \emptyset & \Delta_c(\mathbb{H}_0(c^*)) = \{\mathbb{H}_0(c^*)\} \end{array}$$

The derivated term automaton of E is given in Figure 4.

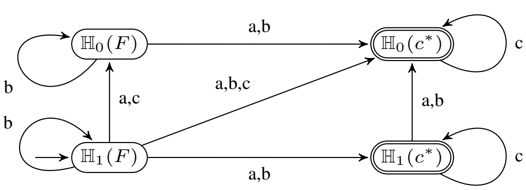

Fig. 4: The derivated term automaton of $E = \mathbb{H}_1(b^*(a+b)c^*)$

(iii) The four first equalities are HLAREs reductions whereas the last one is a HLARE set reduction.

Example 6 *Let $G = \mathbb{L}_1((aba + abb)a(a)^*)$ be the HLARE defined in Example 4 and $w = aba$ be a word in Σ^*. Then:* $\mathrm{r}(aba, G) = \bigvee_{H\in\Delta_{aba}(G)} \mathrm{r}(\varepsilon, H)$. *Let us first compute the HLARE set $\Delta_{aba}(G)$:*

$$\begin{aligned}
\Delta_a(G) &= \{\mathbb{L}_1(baa(a)^*), \mathbb{L}_1(bba(a)^*), \mathbb{L}_0((aba + abb)a(a)^*)\}\\
&= \mathcal{G}_1\\
\Delta_b(\mathcal{G}_1) &= \{\mathbb{L}_1(aa(a)^*), \mathbb{L}_0(baa(a)^*), \mathbb{L}_1(ba(a)^*), \mathbb{L}_0(bba(a)^*), \mathbb{L}_0(a(a)^*)\}\\
&= \mathcal{G}_2\\
\Delta_a(\mathcal{G}_2) &= \{\mathbb{L}_1(a(a)^*), \mathbb{L}_0(aa(a)^*), \mathbb{L}_0((a)^*), \mathbb{L}_0(ba(a)^*), \mathbb{L}_0(a(a)^*)\}\\
&= \mathcal{G}_3
\end{aligned}$$

Hence $\mathrm{r}(aba, G) = \bigvee_{H\in\mathcal{G}_3} \mathrm{r}(\varepsilon, H)$. *Furthermore, since $\varepsilon \in L(\mathbb{L}_0((a)^*))$, there exists an HLARE H in $\mathcal{G}_3$ such that* $\mathrm{r}(\varepsilon, H) = 1$. *Finally, aba belongs to $L(G)$.*

Notice that in this case:

1. *The word w is split up into $s_w = (a, b, \varepsilon, a)$;*
2. *The word $w' = abaa$ in $L((aba + abb)a(a)^*)$ can be split up into $s_{w'} = (a, b, a, a)$;*
3. *It holds $\mathcal{L}(s_w, s_{w'}) = 1$.*

Another split-up is presented in Example 4.

5 Word Comparison Functions, Quotients and Derivatives

In this section, we address the general case. We present two constructions of an automaton from an ARE using Brzozowski's derivatives and Antimirov's ones, respectively leading to a deterministic automaton and a non-deterministic one. We first show how to compute the quotient of a given language $\mathbb{F}_k(L)$ w.r.t. a symbol a, where $\mathbb{F}$ is a given word comparison function, k is an integer and L is a regular language.

5.1 Quotient of a Language

Let $\mathbb{F}$ be a word comparison function associated with a symbol-wise sequence comparison function $\mathcal{F}$ defined over an alphabet Σ. Let k be an integer, a be a symbol in Σ, $u = aw$ be a word of Σ^+, and L' be a regular language over Σ. According to Definition 2, the word u is in $L = \mathbb{F}_k(L')$ if and only if there exists a word $v \in L'$ such that $\mathbb{F}(u, v) \leq k$. According to Definition 1, this is equivalent to the existence of a positive integer n and of an alignment[(iv)] $(u', v') \in \mathrm{Split}_n(u) \times \mathrm{Split}_n(v)$ between u and v, the cost $\mathcal{F}(u', v')$ of which is not greater than k. Let $u' = (u'_1, \ldots, v'_n)$ and $v' = (v'_1, \ldots, v'_n)$. **(a)** If $n = 1$, $\mathbb{F}(u, v) = \mathcal{F}((a), (v'_1))$ and since $u = a$, $a \in L \Leftrightarrow \varepsilon \in \mathbb{F}_{k-\mathcal{F}((a),(v'_1))} {v'_1}^{-1}(L')$. **(b)** Otherwise, let us set $u'' = (u'_2, \ldots, u'_n)$ and $v'' = (v'_2, \ldots, v'_n)$. Moreover, let us set $tu'_2 \cdots u'_n$; let us similarly set $z = v'_2 \cdots v'_n$. Obviously, the word z belongs to ${v'_1}^{-1}(L')$. Since $\mathbb{F}$ is a symbol-wise word comparison function, there exists an alignment (u', v') between u and v satisfying $\mathcal{F}(u', v') \leq k$ if and only if there exists an alignment (u'', v'') between t and z satisfying $\mathcal{F}(u'', v'') \leq k - \mathcal{F}((u'_1), (v'_1))$. According to Definition 1, this is equivalent to the existence of a word $z \in {v'_1}^{-1}(L')$ such that $\mathbb{F}(t, z) \leq k - \mathcal{F}((u'_1), (v'_1))$. According to Definition 2, it is equivalent to say that the word t is in $\mathbb{F}_{k-\mathcal{F}((u'_1),(v'_1))}({v'_1}^{-1}(L'))$. Depending on the value of (u'_1, v'_1) we can distinguish the following cases:

Case 1 $(u'_1, v'_1) = (a, b)$, with $b \in \Sigma$: $u = aw \in L \Leftrightarrow w \in \mathbb{F}_{k-\mathcal{F}(a,b)}(b^{-1}L')$,

(iv) An alignment between two words u and v is a pair (s, s') of sequences of same size such that $s \in \mathrm{Split}(u)$ and $s' \in \mathrm{Split}(v)$.

Case 2 $(u'_1, v'_1) = (a, \varepsilon)$ with $a \in \Sigma$: $u = aw \in L \Leftrightarrow w \in \mathbb{F}_{k-\mathcal{F}(a,\varepsilon)}(L')$,
Case 3 $(u'_1, v'_1) = (\varepsilon, b)$, with $b \in \Sigma$: $u = aw \in L \Leftrightarrow w \in a^{-1}(\mathbb{F}_{k-\mathcal{F}(\varepsilon,b)}(b^{-1}L'))$. Since $w \in a^{-1}\mathbb{F}_k(L') \Leftrightarrow aw \in \mathbb{F}_k(L')$, these three cases provide a recursive expression of the quotient of the language $\mathbb{F}_k(L')$ w.r.t. a symbol $a \in \Sigma$. Unfortunately, its computation may imply a recursive loop, due to Case 3, when $\mathcal{F}((\varepsilon), (b)) = 0$. It is possible to get rid of this loop by precomputing the set of all the quotients of L' w.r.t. words w such that $\mathbb{F}(\varepsilon, w) = 0$. In this purpose, let us set $\mathcal{W}_{\mathcal{F}} = (\bigcup_{b\in\Sigma,\mathcal{F}((\varepsilon),(b))=0}\{b\})^*$ and $X(L') = \{L'\} \cup \bigcup_{w\in\mathcal{W}_{\mathcal{F}}}\{w^{-1}(L')\}$. Notice that if L' is a regular language, the set of its residuals is finite; as a consequence, so is $X(L')$.

Lemma 8 *Let $L = \mathbb{F}_k(L')$ be a language over an alphabet Σ where L' is a regular language, $\mathbb{F}$ is a symbol-wise word comparison function associated with a sequence comparison function $\mathcal{F}$ and a be a symbol in Σ. The* quotient *of L w.r.t. a is the language $a^{-1}(L)$ computed as follows:*

$$a^{-1}(L) = \left\{ \begin{array}{l} \bigcup_{L''\in X(L'),b\in\Sigma}(\mathbb{F}_{k-\mathcal{F}((a),(b))}(b^{-1}(L''))) \cup \bigcup_{L''\in X(L')}\mathbb{F}_{k-\mathcal{F}((a),(\varepsilon))}(L'') \\ \cup\, a^{-1}(\bigcup_{L''\in X(L'),b\in\Sigma,\mathcal{F}((\varepsilon),(b))\neq 0}(\mathbb{F}_{k-\mathcal{F}((\varepsilon),(b))}(b^{-1}(L'')))) \end{array}\right.$$

$$\textit{where } X(L') = \{L'\} \cup \bigcup_{w\in\mathcal{W}_{\mathcal{F}}} w^{-1}(L') \textit{ with } \mathcal{W}_{\mathcal{F}} = (\bigcup_{b\in\Sigma,\mathcal{F}((\varepsilon),(b))=0}\{b\})^*.$$

Proof: For any symbol α, β in $\Sigma \cup \{\varepsilon\}$, let us set $k_{\alpha,\beta} = k - \mathcal{F}((\alpha),(\beta))$.

$$u \in a^{-1}(L) \Leftrightarrow au \in L \Leftrightarrow \exists w \in L', \mathbb{F}(au, w) \in \{0, \ldots, k\}$$

$$\Leftrightarrow \left\{ \begin{array}{l} \quad \exists b \in \Sigma, \exists w_1bw_2 \in L', \mathbb{F}(\varepsilon, w_1) = 0 \wedge \mathbb{F}(u, w_2) \leq k_{a,b} \\ \vee\, \exists w_1w_2 \in L', \mathbb{F}(\varepsilon, w_1) = 0 \wedge \mathbb{F}(u, w_2) \leq k_{a,\varepsilon} \\ \vee\, \exists b \in \Sigma, \exists w_1bw_2 \in L', \mathbb{F}(\varepsilon, w_1) = 0 \wedge \mathcal{F}((\varepsilon),(b)) \neq 0 \wedge \mathbb{F}(au, w_2) \leq k_{\varepsilon,b} \end{array}\right.$$

$$\Leftrightarrow \left\{ \begin{array}{l} \quad \exists b \in \Sigma, \exists w_1 \in \mathcal{W}_{\mathcal{F}}, \exists w_2 \in (w_1b)^{-1}(L'), \mathbb{F}(u, w_2) \leq k_{a,b} \\ \vee\, \exists w_1 \in \mathcal{W}_{\mathcal{F}}, \exists w_2 \in (w_1)^{-1}(L'), \mathbb{F}(u, w_2) \leq k_{a,\varepsilon} \\ \vee\, \exists b \in \Sigma, \exists w_1 \in \mathcal{W}_{\mathcal{F}}, \exists w_2 \in (w_1b)^{-1}L', \\ \quad \mathcal{F}((\varepsilon),(b)) \neq 0 \wedge \mathbb{F}(au, w_2) \leq k_{\varepsilon,b} \end{array}\right.$$

$$\Leftrightarrow \left\{ \begin{array}{l} \quad \exists b \in \Sigma, \exists w_2 \in b^{-1}(\bigcup_{L''\in X(L')} L''), \mathbb{F}(u, w_2) \leq k_{a,b} \\ \vee\, \exists w_2 \in \bigcup_{L''\in X(L')} L'', \mathbb{F}(u, w_2) \leq k_{a,\varepsilon} \\ \vee\, \exists b \in \Sigma, \exists w_2 \in b^{-1}(\bigcup_{L''\in X(L')} L''), \mathcal{F}((\varepsilon),(b)) \neq 0 \wedge \mathbb{F}(au, w_2) \leq k_{\varepsilon,b} \end{array}\right.$$

$$\Leftrightarrow \left\{ \begin{array}{l} \quad \exists b \in \Sigma, u \in \mathbb{F}_{k_{a,b}} \bigcup_{L''\in X(L')} b^{-1}(L'') \\ \vee\, u \in \bigcup_{L''\in X(L')} \mathbb{F}_{k_{a,\varepsilon}}(L'') \\ \vee\, \exists b \in \Sigma, au \in \mathbb{F}_{k_{\varepsilon,b}}(\bigcup_{L''\in X(L')} b^{-1}(L'')) \end{array}\right.$$

$$\Leftrightarrow \left\{ \begin{array}{l} \quad u \in \bigcup_{L''\in X(L'),b\in\Sigma} \mathbb{F}_{k-\mathcal{F}((a),(b))} b^{-1}(L'') \\ \vee\, u \in \bigcup_{L''\in X(L')} \mathbb{F}_{k-\mathcal{F}((a),(\varepsilon))}(L'') \\ \vee\, u \in a^{-1}(\bigcup_{L''\in X(L'),b\in\Sigma,\mathcal{F}((\varepsilon),(b))\neq 0} \mathbb{F}_{k-\mathcal{F}((\varepsilon),(b))} b^{-1}(L'')) \end{array}\right. \qquad \square$$

5.2 Brzozowski Derivatives for an ARE

An extension of Brzozowski derivatives can be directly deduced from the computation of the quotient presented in Lemma 8.

Definition 10 *Let $E = \mathbb{F}_k(E')$ be an ARE over an alphabet Σ where $\mathbb{F}$ is associated with $\mathcal{F}$ and a be a symbol in Σ. The* dissimilar derivative *of E w.r.t. a is the expression $\frac{d'}{d'_a}(E)$ defined by:*

$$\frac{d'}{d'_a}(E) = \left(\begin{array}{l} \quad\sum_{F\in X(E'),b\in\Sigma}(\mathbb{F}_{k-\mathcal{F}((a),(b))}(\frac{d'}{d'_b}(F))) \\ +\sum_{F\in X(E')}\mathbb{F}_{k-\mathcal{F}((a),(\varepsilon))}(F) \\ +\frac{d'}{d'_a}(\sum_{F\in X(E'),b\in\Sigma,\mathcal{F}((\varepsilon),(b))\neq 0}(\mathbb{F}_{k-\mathcal{F}((\varepsilon),(b))}(\frac{d'}{d'_b}(F)))) \end{array} \right)_{\sim_s},$$

where $X(E') = \{E'\} \cup \bigcup_{w\in\mathcal{W}_\mathcal{F}} \frac{d'}{d'_w}(E')$ with $\mathcal{W}_\mathcal{F} = (\bigcup_{b\in\Sigma,\mathcal{F}((\varepsilon),(b))=0}\{b\})^*$.

Let us show that the set of dissimilar derivatives of any HLARE E is finite (Lemma 9), that the dissimilar derivative of E w.r.t. a word w denotes the quotient of $L(E)$ w.r.t. w (Lemma 10) and how to determine whether the empty word belongs to the language denoted by E (Lemma 11).

Lemma 9 *Let $E = \mathbb{F}_k(E')$ be an ARE over an alphabet Σ and $\mathcal{D}_E$ be the set of dissimilar derivatives of E. Then $\mathcal{D}_E$ is a finite set of AREs. Moreover, its computation halts.*

Proof: Consider $\mathbb{F}$ associated with $\mathcal{F}$. Let us show by induction over the structure of E' and by recurrence over k that $\mathcal{D}_E$ is a finite set of AREs.

By induction, the set $\mathcal{D}_{E'}$ is a finite set of AREs. Consequently, since $X(E')$ is a subset of $\mathcal{D}_{E'}$, **(Fact 1)** $X(E')$ is a finite set of derivatives of E'.

In order to show that $\mathcal{D}_E$ is a finite set, let us show that any derivative G of E satisfies the property $\mathrm{P}(E',k)$: G is a finite sum of expressions of type $\mathbb{F}_{k'}(G')$ with $k' \leq k$ and G' a derivative of E'.

According to **Fact 1**, any subexpression $\mathbb{F}_{k-\mathcal{F}((a),(\varepsilon))}(F)$ with $F \in X(E')$ satisfies $\mathrm{P}(E',k)$. Since $X(E')$ is a subset of $\mathcal{D}_{E'}$, $\frac{d'}{d'_b}(F)$ is a derivative of E' for any b in Σ. Consequently, the expression $\sum_{F\in X(E'),b\in\Sigma}(\mathbb{F}_{k-\mathcal{F}((a),(b))}(\frac{d'}{d'_b}(F)))$ also satisfies $\mathrm{P}(E',k)$. Finally, by recurrence hypothesis, for $k' < k$, any derivative of an expression $\mathbb{F}_{k'}(G')$ satisfies $\mathrm{P}(G',k')$. Consequently, any derivative of $\mathbb{F}_{k-\mathcal{F}((\varepsilon),(b))}(\frac{d'}{d'_b}(F))$ satisfies $\mathrm{P}(\frac{d'}{d'_b}(F), k-\mathcal{F}((\varepsilon),(b)))$ if $\mathcal{F}((\varepsilon),(b)) \neq 0$. Since F is a derivative of E', so is $\frac{d'}{d'_b}(F)$, and since $k - \mathcal{F}((\varepsilon),(b)) < k$, any derivative of $\mathbb{F}_{k-\mathcal{F}((\varepsilon),(b))}(\frac{d'}{d'_b}(F))$ satisfies $\mathrm{P}(E',k)$. As a consequence, **(Fact 2)** any derivative of E w.r.t. a symbol a satisfies $\mathrm{P}(E',k)$.

Let us show now that if an expression H satisfies $\mathrm{P}(E',k)$, then any symbol derivative of H also satisfies $\mathrm{P}(E',k)$. Since H is a sum of expressions of type $\mathbb{F}_{k'}(G')$ where $k' \leq k$ and G' is a derivative of E', any symbol derivative H' of H is the sum of the derivatives of the expression H. According to **Fact 2**, any symbol derivative of an expression $\mathbb{F}_{k'}(G')$ satisfies $\mathrm{P}(G',k')$. Since G' is a derivative of E' and $k' \leq k$, any expression satisfying $\mathrm{P}(G',k')$ also satisfies $\mathrm{P}(E',k)$. As a consequence, any derivative of E w.r.t. a word w in Σ^* satisfies $\mathrm{P}(E',k)$.

As a conclusion, since any derivative of E is a sum of expressions all belonging to the finite set $\{\mathbb{F}_{k'}(G) \mid k' \leq k \wedge G \in \mathcal{D}_{E'}\}$, using the ACI-equivalence, $\mathcal{D}_E$ is a finite set of AREs. Moreover, by induction over E' and by recurrence over k, since any derivative of an expression F in $X(E')$ belongs to the finite set of derivatives of E' the computation of which halts, and since $\mathcal{F}((\varepsilon),(b)) \neq 0$ implies that $k - \mathcal{F}((\varepsilon),(b)) < k$, the computation of $\mathcal{D}_E$ halts. $\square$

Lemma 10 *Let $E = \mathbb{F}_k(E')$ be an ARE over an alphabet Σ and a be a symbol in Σ. Then $L(\frac{d'}{d'_a}(E)) = a^{-1}(L(E))$.*

Proof: By induction over the structure of E. According to Lemma 8:

$$a^{-1}(L(E)) = \left\{ \begin{array}{l} \bigcup_{L'' \in X(L(E')), b \in \Sigma} (\mathbb{F}_{k-\mathcal{F}((a),(b))}(b^{-1}(L''))) \\ \cup \bigcup_{L'' \in X(L(E'))} \mathbb{F}_{k-\mathcal{F}((a),(\varepsilon))}(L'') \\ \cup\, a^{-1}(\bigcup_{L'' \in X(L(E')), b \in \Sigma, \mathcal{F}((\varepsilon),(b)) \neq 0} (\mathbb{F}_{k-\mathcal{F}((\varepsilon),(b))}(b^{-1}(L'')))) \end{array} \right. ,$$

where $X(L(E')) = \{L(E')\} \cup \bigcup_{w \in \mathcal{W}_{\mathcal{F}}} w^{-1}(L(E'))$ with $\mathcal{W}_{\mathcal{F}} = (\bigcup_{b \in \Sigma, \mathcal{F}((\varepsilon),(b))=0} \{b\})^*$.

Let $X(E') = \{E'\} \cup \bigcup_{w \in \mathcal{W}_{\mathcal{F}}} \frac{d'}{d'_w}(E')$. By induction over E', for any word w in Σ^*, $w^{-1}(L(E')) = L(\frac{d'}{d'_w}(E'))$. As a consequence, there exists a surjection f from $X(E')$ to $X(L(E'))$ such that for any expression G in $X(E')$, $\mathrm{f}(G) = L(G)$ belongs to $X(L(E'))$. As a consequence:

$$a^{-1}(L(E)) = \left\{ \begin{array}{l} \bigcup_{E'' \in X(E'), b \in \Sigma} (\mathbb{F}_{k-\mathcal{F}((a),(b))}(b^{-1}(L(E'')))) \\ \cup \bigcup_{E'' \in X(E')} \mathbb{F}_{k-\mathcal{F}((a),(\varepsilon))}(L(E'')) \\ \cup\, a^{-1}(\bigcup_{E'' \in X(E'), b \in \Sigma, \mathcal{F}((\varepsilon),(b)) \neq 0} (\mathbb{F}_{k-\mathcal{F}((\varepsilon),(b))}(b^{-1}(L(E''))))) \end{array} \right.$$

By induction over E', for any derivative E'' of E', $b^{-1}(L(E'')) = L(\frac{d'}{d'_b}(E''))$. Consequently:

$$a^{-1}(L(E)) = \left\{ \begin{array}{l} \bigcup_{E'' \in X(E'), b \in \Sigma} (\mathbb{F}_{k-\mathcal{F}((a),(b))}(L(\frac{d'}{d'_b}(E'')))) \\ \cup \bigcup_{E'' \in X(E')} \mathbb{F}_{k-\mathcal{F}((a),(\varepsilon))}(L(E'')) \\ \cup\, a^{-1}(\bigcup_{E'' \in X(E'), b \in \Sigma, \mathcal{F}((\varepsilon),(b)) \neq 0} (\mathbb{F}_{k-\mathcal{F}((\varepsilon),(b))}(L(\frac{d'}{d'_b}(E''))))) \end{array} \right.$$

$$= \left\{ \begin{array}{l} L(\sum_{E'' \in X(E'), b \in \Sigma} (\mathbb{F}_{k-\mathcal{F}((a),(b))}(\frac{d'}{d'_b}(E'')))) \\ \cup\, L(\sum_{E'' \in X(E')} \mathbb{F}_{k-\mathcal{F}((a),(\varepsilon))}(E'')) \\ \cup\, a^{-1}(L(\sum_{E'' \in X(E'), b \in \Sigma, \mathcal{F}((\varepsilon),(b)) \neq 0} (\mathbb{F}_{k-\mathcal{F}((\varepsilon),(b))}(\frac{d'}{d'_b}(E''))))) \end{array} \right.$$

Furthermore, by recurrence over k, for any $\mathcal{F}((\varepsilon),(b)) > 0$, it holds:

$$a^{-1}(L(\mathbb{F}_{k-\mathcal{F}((\varepsilon),(b))}(\frac{d'}{d'_b}(E'')))) = L(\frac{d'}{d'_a}(\mathbb{F}_{k-\mathcal{F}((\varepsilon),(b))}(\frac{d'}{d'_b}(E'')))).$$

Finally,

$$a^{-1}(L(E)) = \left\{ \begin{array}{l} L(\sum_{E'' \in X(E'), b \in \Sigma} (\mathbb{F}_{k-\mathcal{F}((a),(b))}(\frac{d'}{d'_b}(E'')))) \\ \cup\, L(\sum_{E'' \in X(E')} \mathbb{F}_{k-\mathcal{F}((a),(\varepsilon))}(E'')) \\ \cup\, L(\frac{d'}{d'_a}(\sum_{E'' \in X(E'), b \in \Sigma, \mathcal{F}((\varepsilon),(b)) \neq 0} (\mathbb{F}_{k-\mathcal{F}((\varepsilon),(b))}(\frac{d'}{d'_b}(E''))))) \end{array} \right.$$

$$= L(\frac{d'}{d'_a}(E)).$$

□

Lemma 11 *Let $E = \mathbb{F}_k(E')$ be an ARE over an alphabet Σ and a be a symbol in Σ. Let $\mathcal{W}_{\mathcal{F}}$ and $X(E')$ be the sets defined by:*

$$\mathcal{W}_{\mathcal{F}} = (\bigcup_{b \in \Sigma, \mathcal{F}((\varepsilon),(b))=0} \{b\})^* \text{ and}$$
$$X(E') = \{E'\} \cup \bigcup_{w \in \mathcal{W}_{\mathcal{F}}} \frac{d'}{d'_w}(E').$$

Let us consider the language L' defined by:

$$L' = \bigcup_{F \in X(E')} L(F) \cup L(\sum_{F \in X(E'), b \in \Sigma, \mathcal{F}((\varepsilon),(b)) \neq 0} (\mathbb{F}_{k-\mathcal{F}((\varepsilon),(b))}(\frac{d'}{d'_b}(F)))).$$

Then the two following propositions are equivalent:

- $\varepsilon \in L(E)$
- $k \neq \bot \ \wedge\ \varepsilon \in L'$.

Furthermore, this equivalence defines a membership test that halts.

Proof:

Let $\mathcal{W}_{\mathcal{F}} = (\bigcup_{b\in\Sigma,\mathcal{F}((\varepsilon),(b))=0}\{b\})^*$, $X(E') = \{E'\} \cup \bigcup_{w\in\mathcal{W}_{\mathcal{F}}} \frac{d'}{d'_w}(E')$ and for any symbol α, β in Σ, let us set $k_{\alpha,\beta} = k - \mathcal{F}((\alpha),(\beta))$. Obviously, $k = \perp \Rightarrow \varepsilon \notin L(E)$. Consequently, if $k \neq \perp$:

$\varepsilon \in L(E) \Leftrightarrow \exists w \in L(E'), \mathbb{F}(\varepsilon, w) \in \{0,\ldots,k\}$

$$\Leftrightarrow \begin{cases} \quad \exists w \in L(E'), \mathbb{F}(\varepsilon, w) = 0 \\ \vee\ \exists b \in \Sigma, \exists w_1 b w_2 \in L(E'), \\ \quad \mathbb{F}(\varepsilon, w_1) = 0 \wedge \mathcal{F}((\varepsilon),(b)) \neq 0 \wedge \mathbb{F}(\varepsilon, w_2) \leq k_{\varepsilon,b} \end{cases}$$

$$\Leftrightarrow \begin{cases} \quad \exists w \in L(E'), w \in \mathcal{W}_{\mathcal{F}} \\ \vee\ \exists b \in \Sigma, \exists w_1 \in \mathcal{W}_{\mathcal{F}}, \exists w_2 \in (w_1 b)^{-1}(L(E')), \\ \quad \mathcal{F}((\varepsilon),(b)) \neq 0 \wedge \mathbb{F}(\varepsilon, w_2) \leq k_{\varepsilon,b} \end{cases}$$

$$\Leftrightarrow \begin{cases} \quad \exists w \in L(E'), \varepsilon \in w^{-1}(L(F)) \\ \vee\ \exists b \in \Sigma, \exists w_2 \in (b)^{-1}(\bigcup_{F\in X(E')} L(F)), \mathcal{F}((\varepsilon),(b)) \neq 0 \wedge \mathbb{F}(\varepsilon, w_2) \leq k_{\varepsilon,b} \end{cases}$$

$$\Leftrightarrow \begin{cases} \quad \varepsilon \in \bigcup_{F\in X(E')} L(F) \\ \vee\ \exists b \in \Sigma, \exists w_2 \in L(\sum_{F\in X(E')} \frac{d'}{d'_b}(F)), \mathcal{F}((\varepsilon),(b)) \neq 0 \wedge \mathbb{F}(\varepsilon, w_2) \leq k_{\varepsilon,b} \end{cases}$$

$\Leftrightarrow \varepsilon \in \bigcup_{F\in X(E')} L(F) \vee \varepsilon \in L(\sum_{b\in\Sigma, F\in X(E'), \mathcal{F}((\varepsilon),(b))\neq 0} \mathbb{F}_{k_{\varepsilon,b}}(\frac{d'}{d'_b}(F)))$

Furthermore, **(a)** by induction over E', the membership test defined by $\varepsilon \in \bigcup_{F\in X(E')} L(F)$ halts; **(b)** by recurrence over k since $k_{\varepsilon,b} < k$ when $\mathcal{F}((\varepsilon),(b)) \neq 0$, the membership test defined by:

$$\varepsilon \in L(\sum_{F\in X(E'), b\in\Sigma, \mathcal{F}((\varepsilon),(b))\neq 0} (\mathbb{F}_{k-\mathcal{F}((\varepsilon),(b))}(\frac{d'}{d'_b}(F))))$$

halts. $\square$

Lemma 9 ensures that the derivative automaton $B'(E)$ of an ARE E, computed from the set $\mathcal{D}_E$ of dissimilar derivatives of E following the classical way, is a finite recognizer. Lemma 11 ensures that the set of final states can be computed, since the number of derivatives is finite. Finally, Lemma 10 ensures that the DFA D recognizes $L(E)$.

Definition 11 *Let E be an ARE over an alphabet Σ. The tuple $B'(E) = (\Sigma, Q, I, F, \delta)$ is defined by:*

- $Q = \mathcal{D}_E$,
- $I = \{E\}$,
- $F = \{q \in Q \mid \mathrm{r}(\varepsilon, q) = 1\}$,
- $\forall (q,a) \in Q \times \Sigma$, $\delta(q,a) = \{\frac{d'}{d'_a}(q)\}$.

Proposition 3 *Let E be an approximate regular expression. Then:*

$B'(E)$ is a deterministic automaton that recognizes $L(E)$.

Proof: Let $B'(E) = (\Sigma, Q, I, F, \delta)$. Let w be a word in Σ^*. Let us show by recurrence over the length of w that $\delta(E, w) = \{\frac{d'}{d'_w}(E)\}$.

If $w \in \Sigma$, proposition is satisfied by definition of δ.

If $w = w'a$ with $w' \in \Sigma^*$ and $a \in \Sigma$, by recurrence hypothesis it holds $\delta(E, w') = \{\frac{d'}{d'_{w'}}(E)\}$. By definition of δ:

$$\begin{aligned}\delta(E, w'a) &= \delta(\delta(E, w'), a),\\ &= \delta(\{\tfrac{d'}{d'_{w'}}(E)\}, a)\\ &= \delta(\tfrac{d'}{d'_{w'}}(E), a)\\ &= \{\tfrac{d'}{d'_a}(\tfrac{d'}{d'_{w'}}(E))\}\\ &= \{\tfrac{d'}{d'_{w'a}}(E)\}.\end{aligned}$$

As a first consequence, since $\mathrm{Card}(I) = 1$, since δ is a function from $Q \times \Sigma^*$ to 2^Q, and since for any pair (q, a) in $Q \times \Sigma$, $\mathrm{Card}(\delta(q, a)) = 1$, then the tuple $B'(E)$ is a deterministic automaton. Moreover,

$$\begin{aligned}w \in L(B'(E)) &\Leftrightarrow \delta(E, w'a) \cap F \neq \emptyset\\ &\Leftrightarrow \{\tfrac{d'}{d'_{w'}}(E)\} \cap F \neq \emptyset\\ &\Leftrightarrow \tfrac{d'}{d'_{w'}}(E) \in F\\ &\Leftrightarrow \mathrm{r}(\varepsilon, \tfrac{d'}{d'_{w'}}(E)) = 1\\ &\Leftrightarrow \varepsilon \in L(\tfrac{d'}{d'_{w'}}(E))\\ &\Leftrightarrow \varepsilon \in w^{-1}(L(E))\\ &\Leftrightarrow w \in L(E)\end{aligned}$$

□

For any ARE E, the automaton $B'(E)$ is the *dissimilar derivative finite automaton* of E. Consequently, according to Kleene theorem, we have the following corollary.

Corollary 1 *The language denoted by any ARE is regular.*

5.3 Antimirov Derivatives for an ARE

Partial derivatives are defined by means of sets of expressions instead of expressions and thus lead to the construction of a non-deterministic recognizer. We now extend partial derivatives to the family of AREs. For convenience, let us set for $\mathcal{E}$ a set of expressions $\mathbb{F}_k(\mathcal{E}) = \bigcup_{E \in \mathcal{E}} \mathbb{F}_k(E)$ and $L(\mathcal{E}) = \bigcup_{E \in \mathcal{E}} L(E)$.

Definition 12 *Let $E = \mathbb{F}_k(E')$ be an ARE over an alphabet Σ where $\mathbb{F}$ is associated with $\mathcal{F}$ and a be a symbol in Σ. The* partial derivative *of E w.r.t. a is the set $\frac{\partial}{\partial_a}(E)$ computed as follows:*

$$\frac{\partial}{\partial_a}(E) = \left\{ \begin{array}{l} \bigcup_{F \in X(E'), b \in \Sigma} (\mathbb{F}_{k - \mathcal{F}((a),(b))}(\frac{\partial}{\partial_b}(F))) \cup \bigcup_{F \in X(E')} \mathbb{F}_{k - \mathcal{F}((a),(\varepsilon))}(F) \\ \cup \frac{\partial}{\partial_a}(\bigcup_{F \in X(E'), b \in \Sigma, \mathcal{F}((\varepsilon),(b)) \neq 0} (\mathbb{F}_{k - \mathcal{F}((\varepsilon),(b))}(\frac{\partial}{\partial_b}(F)))) \end{array} \right. ,$$

where $\mathcal{W}_{\mathcal{F}} = (\bigcup_{b \in \Sigma, \mathcal{F}((\varepsilon),(b)) = 0} \{b\})^*$ and $X(E') = \{E'\} \cup \bigcup_{w \in \mathcal{W}_{\mathcal{F}}} \frac{\partial}{\partial_w}(E')$.

Lemma 12 *Let $E = \mathbb{F}_k(E')$ be an ARE over an alphabet Σ and a be a symbol in Σ. Then $L(\frac{\partial}{\partial_a}(E)) = a^{-1}(L(E))$.*

Proof:

By induction over the structure of E.

According to Lemma 8:

$$a^{-1}(L(E)) = \begin{cases} \bigcup_{L''\in X(L(E')),b\in\Sigma}(\mathbb{F}_{k-\mathcal{F}((a),(b))}(b^{-1}(L''))) \\ \cup \bigcup_{L''\in X(L(E'))} \mathbb{F}_{k-\mathcal{F}((a),(\varepsilon))}(L'') \\ \cup\, a^{-1}(\bigcup_{L''\in X(L(E')),b\in\Sigma,\mathcal{F}((\varepsilon),(b))\neq 0}(\mathbb{F}_{k-\mathcal{F}((\varepsilon),(b))}(b^{-1}(L'')))) \end{cases},$$

where $X(L(E')) = \{L(E')\} \cup \bigcup_{w\in\mathcal{W}_\mathcal{F}} w^{-1}(L(E'))$ with $\mathcal{W}_\mathcal{F} = (\bigcup_{b\in\Sigma,\mathcal{F}((\varepsilon),(b))=0}\{b\})^*$.

Let $X(E') = \{E'\} \cup \bigcup_{w\in\mathcal{W}_\mathcal{F}} \frac{\partial}{\partial_w}(E')$.

By induction over E', for any word w in Σ^*, $w^{-1}(L(E')) = L(\frac{\partial}{d'_w}(E'))$. As a consequence:

$$\bigcup_{L''\in X(L(E'))} L'' = \bigcup_{E''\in X(E')} L(E'')$$

and:

$$a^{-1}(L(E)) = \begin{cases} \bigcup_{E''\in X(E'),b\in\Sigma}(\mathbb{F}_{k-\mathcal{F}((a),(b))}(b^{-1}(L(E'')))) \\ \cup \bigcup_{E''\in X(E')} \mathbb{F}_{k-\mathcal{F}((a),(\varepsilon))}(L(E'')) \\ \cup\, a^{-1}(\bigcup_{E''\in X(E'),b\in\Sigma,\mathcal{F}((\varepsilon),(b))\neq 0}(\mathbb{F}_{k-\mathcal{F}((\varepsilon),(b))}(b^{-1}(L(E''))))) \end{cases}$$

By induction over E', for any derivative E'' of E', it holds

$$b^{-1}(L(E'')) = L(\frac{\partial}{\partial_b}(E'')).$$

Consequently:

$$a^{-1}(L(E)) = \begin{cases} \bigcup_{E''\in X(E'),b\in\Sigma}(\mathbb{F}_{k-\mathcal{F}((a),(b))}(L(\frac{\partial}{\partial_b}(E'')))) \\ \cup \bigcup_{E''\in X(E')} \mathbb{F}_{k-\mathcal{F}((a),(\varepsilon))}(L(E'')) \\ \cup\, a^{-1}(\bigcup_{E''\in X(E'),b\in\Sigma,\mathcal{F}((\varepsilon),(b))\neq 0}(\mathbb{F}_{k-\mathcal{F}((\varepsilon),(b))}(L(\frac{\partial}{\partial_b}(E'')))) \end{cases}$$

$$= \begin{cases} L(\bigcup_{E''\in X(E'),b\in\Sigma}(\mathbb{F}_{k-\mathcal{F}((a),(b))}(\frac{\partial}{\partial_b}(E'')))) \\ \cup\, L(\bigcup_{E''\in X(E')} \mathbb{F}_{k-\mathcal{F}((a),(\varepsilon))}(E'')) \\ \cup\, a^{-1}(L(\bigcup_{E''\in X(E'),b\in\Sigma,\mathcal{F}((\varepsilon),(b))\neq 0}(\mathbb{F}_{k-\mathcal{F}((\varepsilon),(b))}(\frac{\partial}{\partial_b}(E'')))) \end{cases}$$

Furthermore, by recurrence over k, for any $\mathcal{F}((\varepsilon),(b)) > 0$, it holds:

$$a^{-1}(L(\mathbb{F}_{k-\mathcal{F}((\varepsilon),(b))}(\frac{\partial}{\partial_b}(E'')))) = L(\frac{\partial}{\partial_a}(\mathbb{F}_{k-\mathcal{F}((\varepsilon),(b))}(\frac{\partial}{\partial_b}(E'')))).$$

Finally,

$$a^{-1}(L(E)) = \begin{cases} L(\bigcup_{E''\in X(E'),b\in\Sigma}(\mathbb{F}_{k-\mathcal{F}((a),(b))}(\frac{\partial}{\partial_b}(E'')))) \\ \cup\, L(\bigcup_{E''\in X(E')} \mathbb{F}_{k-\mathcal{F}((a),(\varepsilon))}(E'')) \\ \cup\, L(\frac{\partial}{\partial_a}(\bigcup_{E''\in X(E'),b\in\Sigma,\mathcal{F}((\varepsilon),(b))\neq 0}(\mathbb{F}_{k-\mathcal{F}((\varepsilon),(b))}(\frac{\partial}{\partial_b}(E'')))) \end{cases}$$

and

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

□

Let $\mathcal{DT}_E$ be the set of derivated terms of an ARE E, that is the set of the elements of all the partial derivatives of E.

Lemma 13 *Let $E = \mathbb{F}_k(E')$ be an ARE over an alphabet Σ. Then:*

$$\mathcal{DT}_E \subset \bigcup_{k'\in\{0,\ldots,k\}} \mathbb{F}_{k'}(\mathcal{DT}_{E'}).$$

Moreover, the computation of $\mathcal{DT}_E$ halts.

Proof: Consider that $\mathbb{F}$ is associated with $\mathcal{F}$. Let us define the set $S(E',k) = \bigcup_{k'\in\{0,\ldots,k\}} \mathbb{F}_{k'}(\mathcal{DT}_{E'})$. Let us show by induction over the structure of E' and by recurrence over k that $\mathcal{DT}_E \subset S(E',k)$. Since $X(E')$ is a finite set of derivated terms of E', any subexpression of type $\mathbb{F}_{k-\mathcal{F}((a),(\varepsilon))}(F)$ with $F \in$

$X(E')$ belongs to $S(E',k)$. Since $X(E')$ is a subset of $\mathcal{DT}_{E'}$, $\frac{\partial}{\partial_b}(F)$ is a set of derivated terms of E' for any b in Σ. Consequently, $\bigcup_{F\in X(E'),b\in\Sigma}(\mathbb{F}_{k-\mathcal{F}((a),(b))}(\frac{\partial}{\partial_b}(F)))$ is a subset of $S(E',k)$. Finally, by recurrence hypothesis, for $k'<k$, any partial derivative of an expression $\mathbb{F}_{k'}(H)$ is a subset of $S(H,k')$. Consequently, any partial derivative of $\mathbb{F}_{k-\mathcal{F}((\varepsilon),(b))}(\frac{\partial}{\partial_b}(F))$ is included into $\bigcup_{F'\in\frac{\partial}{\partial_b}(F)} S(F',k-\mathcal{F}((\varepsilon),(b)))$ if $\mathcal{F}((\varepsilon),(b))\neq 0$. Since F is a derivated term of E', so is any expression in $\frac{\partial}{\partial_b}(F)$, and since $k-\mathcal{F}((\varepsilon),(b))\leq k$, any partial derivative of $\mathbb{F}_{k-\mathcal{F}((\varepsilon),(b))}(\frac{\partial}{\partial_b}(F))$ is a subset of $S(E',k)$. As a consequence, **(Fact A)** any derivated term of E w.r.t. a symbol a belongs to $S(E',k)$.

Furthermore, let us show that if $G=\mathbb{F}_{k'}(H)$ is an expression that belongs to $S(E',k)$, then any partial derivative of G is a subset of $S(E',k)$. According to **Fact A**, any partial derivative of an expression $\mathbb{F}_{k'}(H)$ is a subset of $S(H,k')$. When H is a derivated term of E' and $k'\leq k$, any expression in $S(H,k')$ belongs to $S(E',k)$. As a consequence, any derivated term of E belongs to $S(E',k)$.

As a conclusion, $\mathcal{DT}_E\subset S(E',k)=\bigcup_{k'\in\{0,\ldots,k\}}\mathbb{F}_{k'}(\mathcal{DT}_{E'})$. Moreover, by induction over E' and by recurrence over k, since any derivated term of an expression F in $X(E')$ belongs to the finite set of derivated terms of E' the computation of which halts, and since $k-\mathcal{F}((\varepsilon),(b))<k$ when $\mathcal{F}((\varepsilon),(b))\neq 0$, the computation of $\mathcal{DT}_E$ halts. □

Corollary 2 *Let $E=\mathbb{F}_k(E')$ be an ARE over an alphabet Σ. Then $\mathcal{DT}_E$ is a finite set of AREs. Furthermore,* $\mathrm{Card}(\mathcal{DT}_E)\leq\mathrm{Card}(\mathcal{DT}_{E'})\times(k+1)$.

Lemma 14 *Let $E=\mathbb{F}_k(E')$ be an ARE over an alphabet Σ and a be a symbol in Σ. Let $\mathcal{W}_{\mathcal{F}}$ and $X(E')$ be the sets defined by:*

$$\mathcal{W}_{\mathcal{F}}=(\bigcup_{b\in\Sigma,\mathcal{F}((\varepsilon),(b))=0}\{b\})^* \text{ and}$$
$$X(E')=\{E'\}\cup\bigcup_{w\in\mathcal{W}_{\mathcal{F}}}\frac{\partial}{\partial_w}(E').$$

Let L' be the language defined by:

$$L'=\bigcup_{F\in X(E')}L(F)\cup L(\bigcup_{F\in X(E'),b\in\Sigma,\mathcal{F}((\varepsilon),(b))\neq 0}(\mathbb{F}_{k-\mathcal{F}((\varepsilon),(b))}(\tfrac{\partial}{\partial_b}(F)))).$$

Then the two following conditions are equivalent:

- *$\varepsilon\in L(E)$,*
- *$k\neq\perp\wedge\varepsilon\in L'$.*

Furthermore, this equivalence defines a membership test that halts.

Proof: Let $\mathcal{W}_{\mathcal{F}}=(\bigcup_{b\in\Sigma,\mathcal{F}((\varepsilon),(b))=0}\{b\})^*$ and $X(E')=\{E'\}\cup\bigcup_{w\in\mathcal{W}_{\mathcal{F}}}\frac{\partial}{\partial_w}(E')$. For convenience, for any two symbols α and β in $\Sigma\cup\{\varepsilon\}$, let us set $k_{\alpha,\beta}=k-\mathcal{F}((\alpha),(\beta))$. Obviously, if $k=\perp$, $\varepsilon\notin L(E)$. For $k\neq\perp$:

$$\varepsilon\in L(E)\Leftrightarrow\exists w\in L(E'),\mathbb{F}(\varepsilon,w)\in\{0,\ldots,k\}$$

$$\Leftrightarrow\left\{\begin{array}{l}\exists w\in L(E'),\mathbb{F}(\varepsilon,w)=0\\ \vee\ \exists b\in\Sigma,\exists w_1bw_2\in L(E'),\\ \mathbb{F}(\varepsilon,w_1)=0\wedge\mathcal{F}((\varepsilon),(b))\neq 0\wedge\mathbb{F}(\varepsilon,w_2)\leq k_{(\varepsilon),(b))}\end{array}\right.$$

$$\Leftrightarrow\left\{\begin{array}{l}\exists w\in L(E'),w\in\mathcal{W}_{\mathcal{F}}\\ \vee\ \exists b\in\Sigma,\exists w_1\in\mathcal{W}_{\mathcal{F}},\exists w_2\in(w_1b)^{-1}(L(E')),\\ \mathcal{F}((\varepsilon),(b))\neq 0\wedge\mathbb{F}(\varepsilon,w_2)\leq k_{(\varepsilon),(b))}\end{array}\right.$$

$$\Leftrightarrow \left\{ \begin{array}{l} \exists w \in L(E'), \varepsilon \in w^{-1}(\mathcal{W}_{\mathcal{F}}) \\ \vee\, \exists b \in \Sigma, \exists w_2 \in (b)^{-1}(\bigcup_{F \in X(E')} L(F)), \\ \mathcal{F}((\varepsilon),(b)) \neq 0 \wedge \mathbb{F}(\varepsilon, w_2) \leq k_{(\varepsilon),(b))} \end{array} \right.$$

$$\Leftrightarrow \left\{ \begin{array}{l} \varepsilon \in \bigcup_{F \in X(E')} L(F) \\ \vee\, \exists b \in \Sigma, \exists w_2 \in L(\bigcup_{F \in X(E')} \frac{\partial}{\partial_b}(F)), \\ \mathcal{F}((\varepsilon),(b)) \neq 0 \wedge \mathbb{F}(\varepsilon, w_2) \leq k_{(\varepsilon),(b))} \end{array} \right.$$

$$\Leftrightarrow \left\{ \begin{array}{l} \varepsilon \in \bigcup_{F \in X(E')} L(F) \\ \vee\, \varepsilon \in L(\bigcup_{b \in \Sigma, F \in X(E'), \mathcal{F}((\varepsilon),(b)) \neq 0} \mathbb{F}_{k_{(\varepsilon),(b))}}(\frac{\partial}{\partial_b}(F))) \end{array} \right.$$

Furthermore, **(a)** by induction over E', the membership test defined by $\varepsilon \in \bigcup_{F \in X(E')} L(F)$ halts; **(b)** by recurrence over k since $k_{\varepsilon),b} < k$ when $\mathcal{F}((\varepsilon),(b)) \neq 0$, the membership test defined by

$$\varepsilon \in L(\bigcup_{F \in X(E'), b \in \Sigma, \mathcal{F}((\varepsilon),(b)) \neq 0} (\mathbb{F}_{k - \mathcal{F}((\varepsilon),(b))}(\frac{\partial}{\partial_b}(F))))$$

halts. □

Corollary 2 ensures that the derivated term automaton $A(E)$ of an ARE E, computed from the set $\mathcal{DT}_E$ of derivated terms of E following the classical way, is a finite recognizer. Lemma 14 ensures that the set of final states can be computed. Finally, Lemma 12 ensures that the NFA A recognizes $L(E)$.

Definition 13 *Let E be an ARE over an alphabet Σ. The tuple $A(E) = (\Sigma, Q, I, F, \delta)$ is defined by:*

- $Q = \mathcal{DT}_E$,
- $I = \{E\}$,
- $F = \{q \in Q \mid \mathrm{r}(\varepsilon, q) = 1\}$,
- $\forall (q, a) \in Q \times \Sigma$, $\delta(q, a) = \frac{\partial}{\partial_a}(q)$.

Proposition 4 *Let E be an approximate regular expression. Then:*

$A(E)$ is a finite automaton that recognizes $L(E)$.

Proof: Let $A(E) = (\Sigma, Q, I, F, \delta)$. Let w be a word in Σ^*. Let us show by recurrence over the length of w that $\delta(E, w) = \frac{\partial}{\partial_w}(E)$.

If $w \in \Sigma$, proposition is satisfied by definition of δ.

If $w = w'a$ with $w' \in \Sigma^*$ and $a \in \Sigma$, by recurrence hypothesis it holds $\delta(E, w') = \frac{\partial}{\partial_{w'}}(E)$. By definition of δ:

$$\begin{aligned} \delta(E, w'a) &= \delta(\delta(E, w'), a), \\ &= \delta(\frac{\partial}{\partial_{w'}}(E), a) \\ &= \bigcup_{E' \in \frac{\partial}{\partial_{w'}}(E)} \delta(E', a) \\ &= \bigcup_{E' \in \frac{\partial}{\partial_{w'}}(E)} \frac{\partial}{\partial_a}(E') \\ &= \frac{\partial}{\partial_{w'a}}(E) \end{aligned}$$

Consequently,

$$\begin{aligned} w \in L(A(E)) &\Leftrightarrow \delta(E, w'a) \cap F \neq \emptyset \\ &\Leftrightarrow \frac{\partial}{\partial_{w'a}}(E) \cap F \neq \emptyset \\ &\Leftrightarrow \exists E' \in \frac{\partial}{\partial_{w'a}}(E) \mid E' \in F \\ &\Leftrightarrow \exists E' \in \frac{\partial}{\partial_{w'a}}(E) \mid \mathrm{r}(\varepsilon, E') = 1 \\ &\Leftrightarrow \varepsilon \in \bigcup_{E' \in \frac{\partial}{\partial_{w'a}}(E)} L(E') \\ &\Leftrightarrow \varepsilon \in w^{-1}(L(E)) \\ &\Leftrightarrow w \in L(E) \end{aligned}$$

$\square$

For any ARE E, the automaton $A(E)$ is called the *derivated term finite automaton* of E.

5.4 Back to Hamming and Levenshtein Derivation

This subsection is devoted to show the link between HLARE derivation formulae and ARE ones. Given an HLARE E and a word w, the following proposition illustrates the fact that the expression $D'_w(E)$ of Definition 6 (resp. the set of expressions $\Delta_w(E)$ of Definition 8) and the expression $\frac{d'}{d'_w}(E)$ in Definition 10 (resp. the set of expressions $\frac{\partial}{\partial_w}(E)$ in Definition 12) are syntactically equal up to the expression $\emptyset$.

Proposition 5 *Let E be an HLARE over an alphabet Σ. For any symbol a in Σ, the two following conditions are satisfied:*

- $\frac{d'}{d'_a}(E) \in \{(D'_a(E) + \emptyset)_{\sim_s}, D'_a(E)\}$,
- $\frac{\partial}{\partial_a}(E) \in \{\Delta_a(E) \cup \{\emptyset\}, \Delta_a(E)\}$.

Proof: We prove the first membership relation. A similar proof can be given for the second one.

By induction over the structure of an HLARE.

1. If $E = a \in \Sigma$, $E = E_1 + E_2$, $E = E_1 \cdot E_2$ or if $E = E_1^*$, the proposition is trivially checked by similarity of the formulae.
2. If $E = \mathbb{H}_k(E')$, by definition of $\frac{d'}{d'_a}(E)$:

$$\frac{d'}{d'_a}(E) = \begin{pmatrix} \sum_{F \in X(E'), b \in \Sigma} (\mathbb{H}_{k-\mathcal{H}((a),(b))}(\frac{d'}{d'_b}(F))) \\ + \sum_{F \in X(E')} \mathbb{H}_{k-\mathcal{H}((a),(\varepsilon))}(F) \\ + \frac{d'}{d'_a}(\sum_{F \in X(E'), b \in \Sigma, \mathcal{H}((\varepsilon),(b)) \neq 0} (\mathbb{H}_{k-\mathcal{H}((\varepsilon),(b))}(\frac{d'}{d'_b}(F)))) \end{pmatrix}_{\sim_s},$$

where $X(E') = \{E'\} \cup \bigcup_{w \in \mathcal{W}_{\mathcal{H}}} \frac{d'}{d'_w}(E')$ with $\mathcal{W}_{\mathcal{H}} = (\bigcup_{b \in \Sigma, \mathcal{H}((\varepsilon),(b))=0} \{b\})^*$.

By definition of $\mathbb{H}$, $X(E') = \{E'\}$, $\mathcal{H}((a),(b)) \in \{1, 0\}$ and $\mathcal{H}((a),(\varepsilon)) = \perp$ for any two symbols a and b in Σ.

Consequently:

$$\frac{d'}{d'_a}(E) = \begin{pmatrix} \sum_{b \in \Sigma} (\mathbb{H}_{k-\mathcal{H}((a),(b))}(\frac{d'}{d'_b}(E'))) \\ + \sum_{F \in X(E')} \mathbb{H}_{k-\perp}(F) \\ + \frac{d'}{d'_a}(\sum_{F \in X(E'), b \in \Sigma, \mathcal{H}((\varepsilon),(b)) \neq 0} (\mathbb{H}_{k-\perp}(\frac{d'}{d'_b}(F)))) \end{pmatrix}_{\sim_s},$$

and finally

$$\frac{d'}{d'_a}(E) = \begin{pmatrix} \mathbb{H}_k(\frac{d'}{d'_a}(E')) \\ + \sum_{b\in\Sigma\setminus\{a\}}(\mathbb{H}_{k-1}(\frac{d'}{d'_b}(E'))) \\ + \emptyset \\ + \emptyset \end{pmatrix}_{\sim_s}$$

$$= \begin{pmatrix} \mathbb{H}_k(D'_a(E')) \\ + \sum_{b\in\Sigma\setminus\{a\}}(\mathbb{H}_{k-1}(D'_b(E'))) \\ + \emptyset \\ + \emptyset \end{pmatrix}_{\sim_s} \in \{D'_a(E) + \emptyset, D'_a(E)\}.$$

3. If $E = \mathbb{L}_k(E')$, by definition of $\frac{d'}{d'_a}(E)$:

$$\frac{d'}{d'_a}(E) = \begin{pmatrix} \sum_{F\in X(E'),b\in\Sigma}(\mathbb{L}_{k-\mathcal{L}((a),(b))}(\frac{d'}{d'_b}(F))) \\ + \sum_{F\in X(E')}\mathbb{L}_{k-\mathcal{L}((a),(\varepsilon))}(F) \\ + \frac{d'}{d'_a}(\sum_{F\in X(E'),b\in\Sigma,\mathcal{L}((\varepsilon),(b))\neq 0}(\mathbb{L}_{k-\mathcal{H}((\varepsilon),(b))}(\frac{d'}{d'_b}(F)))) \end{pmatrix}_{\sim_s},$$

where $X(E') = \{E'\} \cup \bigcup_{w\in\mathcal{W}_\mathcal{L}} \frac{d'}{d'_w}(E')$ with $\mathcal{W}_\mathcal{L} = (\bigcup_{b\in\Sigma,\mathcal{L}((\varepsilon),(b))=0}\{b\})^*$.

By definition of $\mathbb{L}$, $X(E') = \{E'\}$, $\mathcal{L}((a),(b)) \in \{1,0\}$ and $\mathcal{L}((a),(\varepsilon)) = \mathcal{L}((\varepsilon,(a))) = 1$ for any two symbols a and b in Σ.

Consequently:

$$\frac{d'}{d'_a}(E) = \begin{pmatrix} \sum_{b\in\Sigma}(\mathbb{L}_{k-\mathcal{L}((a),(b))}(\frac{d'}{d'_b}(E'))) \\ + \mathbb{L}_{k-1}(E') \\ + \frac{d'}{d'_a}(\sum_{b\in\Sigma}(\mathbb{L}_{k-1}(\frac{d'}{d'_b}(E')))) \end{pmatrix}_{\sim_s}$$

$$= \begin{pmatrix} \mathbb{L}_k(\frac{d'}{d'_a}(E'))) \\ + \sum_{b\in\Sigma}(\mathbb{L}_{k-1}(\frac{d'}{d'_b}(E'))) \\ + \mathbb{L}_{k-1}(E') \\ + \frac{d'}{d'_a}(\sum_{b\in\Sigma}(\mathbb{L}_{k-1}(\frac{d'}{d'_b}(E')))) \end{pmatrix}_{\sim_s}$$

Finally, by induction hypothesis and by recurrence over k,

$$\frac{d'}{d'_a}(E) = \begin{pmatrix} \mathbb{L}_k(D'_a(E'))) \\ + \sum_{b\in\Sigma}(\mathbb{L}_{k-1}(D'_b(E'))) \\ + \mathbb{L}_{k-1}(E') \\ + D'_a(\sum_{b\in\Sigma}(\mathbb{L}_{k-1}(D'_b(E')))) \end{pmatrix}_{\sim_s} = D'_a(E).$$

□

As a corollary of Proposition 5, the proofs of the lemmas and propositions of Section 4 can be deduced from the corresponding ones of Section 5.

6 Conclusion

The similarity operators that equip the family of approximate regular expressions make AREs to be a nice tool to deal with approximate regular expression matching. The extension of dissimilar derivatives and partial derivatives to the family of AREs allows us to provide a syntactical solution to the approximate membership problem; moreover in each case the set of derivatives is finite and thus this extension also yields the construction of a recognizer. An additional advantage of similarity operators is that they can be combined with other regular operators, such as intersection and complementation operators [4], in order to produce even smaller expressions.

References

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

[2] Ricardo A. Baeza-Yates and Gaston H. Gonnet. Fast text searching for regular expressions or automaton searching on tries. *J. Assoc. Comput. Mach.*, 43(6):915–936, 1996.

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

[4] P. Caron, J.-M. Champarnaud, and L. Mignot. Partial derivatives of an extended regular expression. In Adrian Horia Dediu, Shunsuke Inenaga, and Carlos Martín-Vide, editors, *LATA*, volume 6638 of *Lecture Notes in Computer Science*, pages 179–191. Springer, 2011.

[5] J.-M. Champarnaud, H. Jeanne, and L. Mignot. Approximate regular expressions and their derivatives. In Adrian Horia Dediu and Carlos Martín-Vide, editors, *LATA*, volume 7183 of *Lecture Notes in Computer Science*, pages 179–191. Springer, 2012.

[6] M. Crochemore and T. Lecroq. Text searching and indexing. In Z. Ésik, C. Martín-Vide, and V. Mitrana, editors, *Recent Advances in Formal Languages and Applications*, volume 25 of *Studies in Computational Intelligence*, pages 43–80. Springer, 2006.

[7] N. El-Mabrouk. *Recherche approchée de motifs - Application à des séquences biologiques structurées*. PhD thesis, LITP, Université Paris 7, France, 1996.

[8] J. Holub. Dynamic programming - NFA simulation. In J.-M. Champarnaud and D. Maurel, editors, *CIAA*, volume 2608 of *Lecture Notes in Computer Science*, pages 295–300. Springer, 2002.

[9] P. Jokinen, J. Tarhio, and E. Ukkonen. A comparison of approximate string matching algorithms. *Softw., Pract. Exper.*, 26(12):1439–1458, 1996.

[10] S. Kleene. Representation of events in nerve nets and finite automata. *Automata Studies*, Ann. Math. Studies 34:3–41, 1956. Princeton U. Press.

[11] P. Muzátko. Approximate regular expression matching. In *Stringology*, pages 37–41. Department of Computer Science and Engineering, Faculty of Electrical Engineering, Czech Technical University, 1996.

[12] Eugene Myers and Webb Miller. Approximate matching of regular expressions. *Bull. Math. Biol.*, 51:5–37, 1989. 10.1007/BF02458834.

[13] J. Myhill. Finite automata and the representation of events. *WADD*, TR-57-624:112–137, 1957.

[14] G. Navarro. A guided tour to approximate string matching. *ACM Computing Surveys*, 33(1):31–88, 2001.

[15] G. Navarro. Approximate regular expression matching. In M.-Y. Kao, editor, *Encyclopedia of Algorithms*, pages 46–48. Springer, 2008.

[16] A. Nerode. Linear automata transformation. In *Proceedings of AMS 9*, pages 541–544, 1958.

[17] K. U. Schulz and S. Mihov. Fast string correction with levenshtein automata. *IJDAR*, 5(1):67–85, 2002.

[18] E. Ukkonen and D. Wood. Approximate string matching with suffix automata. *Algorithmica*, 10:353–364, 1993. 10.1007/BF01769703.

[19] Sun Wu, Udi Manber, and Eugene W. Myers. A subquadratic algorithm for approximate regular expression matching. *J. Algorithms*, 19(3):346–360, 1995.