Manipulation of Extended Regular Expressions with Derivatives

Rafaela Bastos Nelma Moreira Rogério Reis
CMUP & DCC, Faculdade de Ciências da Universidade do Porto
Rua do Campo Alegre, 4169-007 Porto, Portugal

Technical Report Series: DCC-2013-11
Version 1.0 September 2013

Departamento de Ciência de Computadores
&
Laboratório de Inteligência Artificial e Ciência de Computadores

Faculdade de Ciências da Universidade do Porto
Rua do Campo Alegre, 1021/1055,
4169-007 PORTO,
PORTUGAL

Tel: 220 402 900 Fax: 220 402 950
http://www.dcc.fc.up.pt/Pubs/

Manipulation of Extended Regular Expressions with Derivatives*

Rafaela Bastos, Nelma Moreira, Rogério Reis
up200801458@alunos.dcc.fc.up.pt ,{nam,rvr}@dcc.fc.up.pt
CMUP & DCC-FC, Universidade do Porto
Rua do Campo Alegre 687, 4169-007 Porto, Portugal

DCC-FC, Universidade do Porto
Rua do Campo Alegre 1021/1055, 4169-007 Porto, Portugal

Abstract

The use of derivatives for efficiently deciding equivalence and membership in regular languages has been a major topic of recent research. To ensure termination, regular expressions must be considered modulo some algebraic properties such as associativity, commutativity, and idempotence of union (ACI). In this paper we describe an implementation of regular expressions modulo ACI and several derivative based methods within the FAdo system. Although regular languages are trivially closed for boolean operations, the manipulation of intersection and complementation with regular expressions or non-deterministic finite automata is non trivial and leads to an exponential blow up. However, due to several applications where extended regular expressions (XRE) are used to represent information, it is important the extension of derivative based methods to those operations. Continuing work of Caron et al., we present new algorithms for computing the (extended) equation automaton and deciding membership and equivalence of XRE using (partial) derivatives.

1 Extended Regular Expressions and Kleene Algebra

In this section we briefly review some basic definitions about extended regular expression and Kleene Algebra.

Let $\Sigma = \{\sigma_1, \dots, \sigma_k\}$ be an *alphabet* of size k. A *word* over an alphabet Σ is a finite sequence of symbols of Σ. The *empty word* is denoted by ϵ. The set Σ^* is the set of all words over Σ. A *language* over Σ is a subset of Σ^*. The set R of *extended regular expressions* (ERE) over an alphabet Σ is defined by:

$$\alpha := \emptyset \mid \epsilon \mid \Sigma^+ \mid \Sigma^+ \mid \sigma_1 \mid \dots \mid \sigma_n \mid (\alpha + \beta) \mid (\alpha.\beta) \mid \alpha^* \mid (\alpha \cap \beta) \mid \neg\alpha$$

where Σ^+ and Σ^* are equivalent to $\neg\epsilon$ and $\neg\emptyset$, respectively, and the operator . (concatenation) is often omitted.

*This work was partially funded by the European Regional Development Fund through the programme COMPETE and by the Portuguese Government through the FCT under projects PEst-C/MAT/UI0144/2011 and CANTE-PTDC/EIA-CCO/101904/2008.

The language $L(\alpha)$ associated to α, is inductively defined as follows:

$$\begin{array}{lcllcl} L(\emptyset) & = & \emptyset & L(\alpha+\beta) & = & L(\alpha)\cup L(\beta) \\ L(\epsilon) & = & \{\epsilon\} & L(\alpha\cap\beta) & = & L(\alpha)\cap L(\beta) \\ L(\Sigma^*) & = & \Sigma^*\backslash L(\emptyset) & L(\alpha\beta) & = & L(\alpha).L(\beta) \\ L(\Sigma^+) & = & \Sigma^*\backslash L(\epsilon) & L(\alpha^*) & = & L(\alpha)^* \\ L(\sigma) & = & L(\sigma) & L(\neg\alpha) & = & \Sigma^*\backslash L(\alpha) \end{array}$$

where $\sigma \in \Sigma$.

An extended regular expression represents the empty word ϵ if and only if:

$$\begin{array}{lcllcl} \epsilon(\emptyset) & = & \emptyset & \epsilon(\alpha\cap\beta) & = & \epsilon(\alpha)\cap\epsilon(\beta) \\ \epsilon(\epsilon) & = & \epsilon & \epsilon(\alpha\beta) & = & \epsilon(\alpha)\epsilon(\beta) \\ \epsilon(\Sigma^*) & = & \epsilon & \epsilon(\alpha^*) & = & \epsilon \\ \epsilon(\Sigma^+) & = & \emptyset & & & \\ \epsilon(\sigma) & = & \emptyset & \epsilon(\neg\alpha) & = & \begin{cases}\epsilon, & \text{if } \epsilon(\alpha)\neq\epsilon \\ \emptyset, & \text{otherwise}\end{cases} \\ \epsilon(\alpha+\beta) & = & \epsilon(\alpha)+\epsilon(\beta) & & & \end{array}$$

We say that two regular expressions α and β are equivalent, $\alpha \equiv \beta$, if they represent the same language, i.e, $L(\alpha) = L(\beta)$.

Let $\sigma, \sigma_1, \ldots, \sigma_n$ be symbols of the alphabet Σ and α, β and γ be extended regular expressions. From the axiom sytem AX in [AM94] and the axiom system F in [Sal66], we have the equivalences:

$$\alpha + (\beta + \gamma) \equiv (\alpha + \beta) + \gamma \qquad (A_1)$$
$$(\alpha.\beta).\gamma \equiv \alpha.(\beta.\gamma) \qquad (A_2)$$
$$\alpha + \beta \equiv \beta + \alpha \qquad (A_3)$$
$$\alpha.(\beta + \gamma) \equiv \alpha.\beta + \alpha.\gamma \qquad (A_4)$$
$$(\alpha + \beta).\gamma \equiv \alpha.\gamma + \beta.\gamma \qquad (A_5)$$
$$\alpha + \alpha \equiv \alpha \qquad (A_6)$$
$$\alpha.\epsilon \equiv \alpha \qquad (A_7)$$
$$\alpha.\emptyset \equiv \emptyset \qquad (A_8)$$
$$\alpha + \emptyset \equiv \alpha \qquad (A_9)$$
$$\epsilon + \alpha.\alpha^* \equiv \alpha^* \qquad (A_{10})$$
$$(\epsilon + \alpha)^* \equiv \alpha^* \qquad (A_{11})$$
$$\epsilon \cap (\alpha.\beta) \equiv (\epsilon \cap \alpha) \cap \beta \qquad (A_{12})$$
$$\epsilon \cap \alpha^* \equiv \epsilon \qquad (A_{13})$$
$$\epsilon \cap \sigma \equiv \emptyset \qquad (A_{14})$$
$$\emptyset \cap \alpha \equiv \emptyset \qquad (A_{15})$$
$$\alpha \cap \alpha \equiv \alpha \qquad (A_{16})$$
$$\alpha \cap \beta \equiv \beta \cap \alpha \qquad (A_{17})$$
$$\alpha \cap (\beta \cap \gamma) \equiv (\alpha \cap \beta) \cap \gamma \qquad (A_{18})$$
$$\alpha \cap (\beta + \gamma) \equiv (\alpha \cap \beta) + (\alpha \cap \gamma) \qquad (A_{19})$$
$$\alpha + (\alpha \cap \beta) \equiv \alpha \qquad (A_{20})$$
$$(\sigma_1.\alpha) \cap (\sigma_2.\beta) \equiv (\sigma_1 \cap \sigma_2).(\alpha \cap \beta) \qquad (A_{21})$$
$$(\alpha.\sigma_1) \cap (\beta.\sigma_2) \equiv (\alpha \cap \beta).(\sigma_1 \cap \sigma_2) \qquad (A_{22})$$
$$\sigma_i \cap \sigma_j \equiv \emptyset \quad \forall \sigma_i \neq \sigma_j \qquad (A_{23})$$

$$(\neg\alpha \cap \neg\beta) \equiv \neg\alpha + \neg\beta \qquad (A_{24})$$
$$(\neg\alpha + \neg\beta) \equiv \neg\alpha \cap \neg\beta \qquad (A_{25})$$

The *derivative* of an extended regular expression α with respect to a symbol $\sigma \in \Sigma$, written $d_\sigma(\alpha)$, is a regular expression such that:

$$L(d_\sigma(\alpha)) = \{w \mid \sigma w \in L(\alpha)\}$$

And the inductively definition is the following:

$$\begin{aligned}
&d_\sigma(\emptyset) = \emptyset\\
&d_\sigma(\epsilon) = \emptyset\\
&d_\sigma(\sigma) = \epsilon\\
&d_\sigma(\sigma') = \emptyset\\
&d_\sigma(\alpha + \beta) = d_\sigma(\alpha) + d_\sigma(\beta)\\
&d_\sigma(\alpha \cap \beta) = d_\sigma(\alpha) \cap d_\sigma(\beta)\\
&d_\sigma(\alpha\beta) = d_\sigma(\alpha)\beta + \epsilon(\alpha)d_\sigma(\beta)\\
&d_\sigma(\alpha^*) = d_\sigma(\alpha)\alpha^*\\
&d_\sigma(\neg\alpha) = \neg d_\sigma(\alpha)
\end{aligned}$$

Two extended regular expressions are *modulo-aci* if one can be transformed to the other by using the aci-rules, ie, applying the associatitivity, commutativity and idempotence of the intersection ($\cap$) (Axioms A_{18}, A_{17} and A_{16}) and disjuction ($+$) (Axioms A_1, A_3 and A_6) operators, and applying the associativity of the concatenation (.) (Axiom A_2). Brzozowski proved in [Brz64] that the set of derivatives of a regular expression being a finite set is enough that is modulo-aci.

The set o *partial derivative* of a non-extended regular expression w.r.t. a symbol $\sigma \in \Sigma$, denoted by $\partial_\sigma(\alpha)$, is the set of regular expressions defined as follows:

$$\begin{aligned}
&\partial_\sigma(\emptyset) = \emptyset\\
&\partial_\sigma(\epsilon) = \emptyset\\
&\partial_\sigma(\sigma) = \{\epsilon\}\\
&\partial_\sigma(\sigma') = \emptyset\\
&\partial_\sigma(\alpha + \beta) = \partial_\sigma(\alpha) \cup \partial_\sigma(\beta)\\
&\partial_\sigma(\alpha\beta) = \partial_\sigma(\alpha)\beta \cup \epsilon(\alpha)\partial_\sigma(\beta)\\
&\partial_\sigma(\alpha^*) = \partial_\sigma(\alpha)\alpha^*
\end{aligned}$$

2 Derivatives

In [PCM11] Caron et al. describe a definition for the partial derivative of an extended regular expression, though this definition requires that the regular expression must be in disjunction normal form. In this section, a new definition for partial derivative is described that not requires such assumption.

Let $\Sigma = \{\sigma_1, \ldots, \sigma_k\}$ be an *alphabet* of size k. Consider the following for represent an *extended regular expressions* (ERE) over the alphabet Σ:

$$\alpha := \emptyset \mid \epsilon \mid \Sigma^+ \mid \Sigma^+ \mid \sigma_1 \mid \ldots \mid \sigma_n \mid [\alpha, \ldots, \alpha] \mid \alpha.\alpha \mid \alpha^* \mid \langle \alpha, \ldots, \alpha \rangle\rangle \mid \neg\alpha$$

Where $[\alpha_1, \ldots, \alpha_n]$ and $\langle \alpha_1, \ldots, \alpha_n \rangle$ represent the regular expressions $(\alpha_1 + \ldots + \alpha_n)$ and $(\alpha_1 \cap \ldots \cap \alpha_n)$, respectively.

Definition 1. *Let α be an extended regular expression and σ a symbol in Σ, the partial derivative of α w.r.t. σ, $\partial_\sigma(\alpha)$is defined inductively as follows:*

$$\begin{array}{lcl@{\qquad\qquad}lcl}
\partial_\sigma(\Sigma^*) & = & \{\Sigma^*\} & \partial_\sigma([e_1, \ldots, e_n]) & = & \{f \mid f \in \bigcup \partial_\sigma(e_i)\} \\
\partial_\sigma(\Sigma^+) & = & \{\Sigma^*\} & \partial_\sigma(\langle e_1, \ldots, e_n \rangle) & = & \{\langle f_1, \ldots, f_n \rangle \mid f_i \in \partial_\sigma(e_i)\} \\
\partial_\sigma(\emptyset) & = & \emptyset & & & \\
\partial_\sigma(\epsilon) & = & \emptyset & \partial_\sigma(e_1 e_2) & = & \begin{cases} \{f_1 e_2 \mid f_1 \in \partial_\sigma(e_1)\} \cup \partial_\sigma(e_2), \text{if } \epsilon(e_1) = \epsilon \\ \{f_1.e_2 \mid f_1 \in \partial_\sigma(e_1)\}, \text{otherwise} \end{cases} \\
\partial_\sigma(\sigma) & = & \{\epsilon\} & \partial_\sigma(e^*) & = & \{f e^* \mid f \in \partial_\sigma(e)\} \\
\partial_\sigma(\sigma') & = & \emptyset & \partial_\sigma(\neg e) & = & \overline{\partial_\sigma}(e)
\end{array}$$

$$\begin{array}{lcl@{\qquad\qquad}lcl}
\overline{\partial_\sigma}(\Sigma^*) & = & \emptyset & \overline{\partial_\sigma}([e_1, \ldots, e_n]) & = & \{\langle \neg f \mid f \in \bigcup \partial_\sigma(e_i) \rangle\} \\
\overline{\partial_\sigma}(\Sigma^+) & = & \emptyset & \overline{\partial_\sigma}(\langle e_1, \ldots, e_n \rangle) & = & \{\langle \neg f \mid f \in \bigcup \partial_\sigma(e_i) \rangle \ \mid \forall i\} \\
\overline{\partial_\sigma}(\emptyset) & = & \{\Sigma^*\} & \overline{\partial_\sigma}(e_1 e_2)) & = & \{\langle \neg f \mid f \in \partial_\sigma(e_1 e_2) \rangle\} \\
\overline{\partial_\sigma}(\epsilon) & = & \{\Sigma^*\} & \overline{\partial_\sigma}(e^*) & = & \{\langle \neg f \mid f \in \partial_\sigma(e^*) \rangle\} \\
\overline{\partial_\sigma}(\sigma) & = & \{\Sigma^+\} & \overline{\partial_\sigma}(\neg e) & = & \partial_\sigma(e) \\
\overline{\partial_\sigma}(\sigma') & = & \{\Sigma^*\} & & &
\end{array}$$

Lemma 1. *Let α be an extended regular expression and $\sigma \in \Sigma$, thus*

$$L(d_\sigma(\neg\alpha)) = \Sigma^* \backslash d_\sigma(\alpha) \tag{2.1}$$

Proof. According to the definition of derivatives, we have:

$$\begin{aligned}
L(d_\sigma(\neg\alpha)) &= \{w \mid \sigma w \in L(\neg\alpha)\} \\
&= \{w \mid \sigma w \notin L(\alpha)\} \\
&= \Sigma^* \backslash \{w \mid \sigma w \in L(\alpha)\} \\
&= \Sigma^* \backslash d_\sigma(\alpha)
\end{aligned}$$

□

Proposition 1. *Let α be an extended regular expression over an alphabet Σ and σ a symbol in Σ, then*

$$L(\partial_\sigma(\neg\alpha)) = \Sigma^* \backslash \partial_\sigma(\alpha) \tag{2.2}$$

Proof. By induction on the structure of the extended regular expression α:

For $\alpha \equiv \Sigma^*$:

$$L(\overline{\partial_\sigma}(\Sigma^*)) = L(\overline{\partial_\sigma}(\neg\emptyset)) = L(\partial_\sigma(\emptyset)) = L(\emptyset) = \emptyset.$$

If α is Σ^+:

$$L(\overline{\partial_\sigma}(\Sigma^+)) = L(\overline{\partial_\sigma}(\neg\epsilon)) = L(\partial_\sigma(\epsilon)) = L(\emptyset) = \emptyset.$$

If α is $\emptyset$, ϵ or σ':

$$\begin{aligned} L(\overline{\partial_\sigma}(\alpha)) &= L(\{\Sigma^*\}) = \Sigma^* \\ &= \Sigma^* \backslash \emptyset = \Sigma^* \backslash \partial_\sigma(\alpha). \end{aligned}$$

If α is σ

$$L(\overline{\partial_\sigma}(\sigma) = L(\{\Sigma^+\}) = \Sigma^+ = \Sigma^* \backslash \{\epsilon\} = \Sigma^* \backslash \partial_\sigma(\sigma).$$

Consider that α is $[\alpha_1, \ldots, \alpha_n]$, where α_i is an extended regular expression, for $i = 1, \ldots, n$:

$$\begin{aligned} L(\overline{\partial_\sigma}([\alpha_1, \ldots, \alpha_n])) &= L(\{\langle \neg\alpha' \mid \exists \alpha' \in \partial_\sigma(\alpha_i)\rangle\}) \\ &= L(\{\neg[\alpha' \mid \exists \alpha' \in \partial_\sigma(\alpha_i)]\}) \\ &= \Sigma^* \backslash \{\alpha' \mid \exists \alpha' \in \partial_\sigma(\alpha_i)\} \\ &= \Sigma^* \backslash \partial_\sigma([\alpha_1, \ldots, \alpha_n]). \end{aligned}$$

If α is $\langle \alpha_1, \ldots, \alpha_n \rangle$:

$$\begin{aligned} L(\overline{\partial_\sigma}(\langle \alpha_1, \ldots, \alpha_n \rangle) &= L(\{\langle \neg\alpha' \mid \alpha' \in \bigcup \partial_\sigma(e_i)\rangle \ \mid \forall i\}) \\ &= L(\langle \neg\alpha' \mid \alpha' \in \partial_\sigma(\alpha_1)\rangle) \cup \ldots \cup L(\langle \neg\alpha' \mid \alpha' \in \partial_\sigma(\alpha_n)\rangle) \\ &= L(\neg[\alpha' \mid \alpha' \in \partial_\sigma(\alpha_1)]) \cup \ldots \cup L(\neg[\alpha' \mid \alpha' \in \partial_\sigma(\alpha_n)]) \\ &= \Sigma^* \backslash L(\{\alpha' \mid \alpha' \in \partial_\sigma(\alpha_1)\}) \cup \ldots \cup \Sigma^* \backslash L(\{[\alpha' \mid \alpha' \in \partial_\sigma(\alpha_n)\}) \\ &= \Sigma^* \backslash (L(\{\alpha' \mid \alpha' \in \partial_\sigma(\alpha_1)\}) \cap \ldots \cap L(\{[\alpha' \mid \alpha' \in \partial_\sigma(\alpha_n)\})) \\ &= \Sigma^* \backslash \{\langle \alpha'_1, \ldots, \alpha'_n \rangle \mid \alpha'_i \in \partial(\alpha_i)\} \\ &= \Sigma^* \backslash \partial_\sigma(\langle \alpha_1, \ldots, \alpha_n \rangle). \end{aligned}$$

If α is $\alpha_1\alpha_2$:

$$\begin{aligned} L(\overline{\partial_\sigma}(\alpha_1\alpha_2)) &= L(\{\langle \neg\alpha' \mid \alpha' \in \partial_\sigma(\alpha_1\alpha_2)\rangle\}) \\ &= L(\{\neg[\alpha' \mid \alpha' \in \partial_\sigma(\alpha_1\alpha_2)]\} \\ &= \Sigma^* \backslash L(\{\alpha' \mid \alpha' \in \partial_\sigma(\alpha_1\alpha_2)\}) \\ &= \Sigma^* \backslash L(\partial_\sigma(\alpha_1\alpha_2)). \end{aligned}$$

If α is β^*, where β is an extended regular expression

$$\begin{aligned} L(\overline{\partial_\sigma}(\beta^*)) &= L(\{\langle \neg\alpha' \mid \alpha' \in \partial_\sigma(\beta^*)\rangle\}) \\ &= L(\{\neg[\alpha' \mid \alpha' \in \partial_\sigma(\beta^*)]\} \\ &= \Sigma^* \backslash L(\{\alpha' \mid \alpha' \in \partial_\sigma(\beta^*)\}) \\ &= \Sigma^* \backslash L(\partial_\sigma(\beta^*)). \end{aligned}$$

If α is $\neg\beta$

$$L(\overline{\partial_\sigma}(\neg\beta)) = L(\partial_\sigma(\neg\neg\beta)) = L(\partial_\sigma(\beta)).$$

$\square$

Example 1. *Let* $\alpha = \neg(a^* \cap \neg ba)$ *be an extended regular expression with* $\Sigma = \{a, b\}$*, where* α *is represented in the implementation by* $\neg\langle a^*, \neg ba\rangle$*. The correspondent partial derivative* $\partial_a(\alpha)$ *is calculate as follows:*

$$\begin{aligned}\partial_a(\neg\langle a^*, \neg ba\rangle) &= \overline{\partial_a}(\neg\langle a^*, \neg ba\rangle)\\ &= \{\langle \neg f \mid f \in \partial_a(a^*)\rangle\} \cup \{\langle \neg f \mid f \in \partial_a(\neg ba)\rangle\}\end{aligned}$$

Since,

$$\partial_a(a^*) = \{a^*\}$$

and

$$\begin{aligned}\partial_a(\neg ba) &= \{f_1 a \mid f_1 \in \partial_a(\neg b)\} \cup \partial_a(a)\\ &= \{f_1 a \mid f_1 \in \overline{\partial_a}(b)\} \cup \partial_a(a)\\ &= \{\Sigma^* a\} \cup \{\epsilon\}\\ &= \{\Sigma^* a, \epsilon\},\end{aligned}$$

because $\epsilon(\neg b) = \epsilon$*. We have:*

$$\begin{aligned}\partial_a(\neg\langle a^*, \neg ba\rangle) &= \{\langle \neg f \mid f \in \{a^*\}\rangle\} \cup \{\langle \neg f \mid f \in \{\Sigma^* a, \epsilon\}\rangle\}\\ &= \{a^*, \langle \Sigma^* a, \epsilon\rangle\}\end{aligned}$$

Champarnaud and Ziadi [CZ01] showed that partial derivatives and Mirkin's prebases [Mir66] lead to identical constructions of non-deterministic automata. Here, we give an extended version for intersection of the algorithm.

Let α_0 be a regular expression. A set $\pi(\alpha_0) = \{\alpha_1, \ldots, \alpha_n\}$, where $\alpha_1, \ldots, \alpha_n$ are non-empty regular expressions, is called a *support* of α_0 if, for $i = 0, \ldots, n$, there are $\alpha_{il} \in R$ ($l = 1, \ldots, k$), linear combinations of the elements in $\pi(\alpha_0)$, such that $\alpha_i = \sigma_1.\alpha_{i1} + \ldots + \sigma_k.\alpha_{ik} + \epsilon(\alpha_i)$, where, as above, $\Sigma = \{\sigma_1, \ldots, \sigma_k\}$ is the considered alphabet. If $\pi(\alpha)$ is a support pf α, then the set $\pi(\alpha) \cup \{\alpha\}$ is called a *prebase* of α.

Proposition 2. *Let* α *be an extended regular expression and* σ *a symbol of the alphabet, then the set* $\pi(\alpha)$*, inductively defined by*

$$\begin{aligned}\pi(\Sigma^*) &= \{\Sigma^*\}\\ \pi(\Sigma^+) &= \{\Sigma^*\}\\ \pi(\emptyset) &= \emptyset\\ \pi(\epsilon) &= \emptyset\\ \pi(\sigma) &= \{\epsilon\}\\ \pi(\alpha + \beta) &= \pi(\alpha) \cup \pi(\beta)\\ \pi(\alpha \cap \beta) &= \{\alpha' \cap \beta' \mid \alpha' \in \pi(\alpha), \beta' \in \pi(\beta)\}\\ \pi(\alpha\beta) &= \{\alpha'\beta \mid \beta' \in \pi(\alpha)\} \cup \pi(\beta)\\ \pi(\alpha^*) &= \{\alpha'\alpha^* \mid \alpha' \in \pi(\alpha)\}\end{aligned}$$

is a support of α*.*

Proof. In [CZ01] is the proof for all definitions except for $\pi(\beta \cap \gamma)$, which is proved below.

Let $\pi(\beta_0) = \{\beta_1, \ldots, \beta_n\}$ and $\pi(\gamma_0) = \{\gamma_1, \ldots, \gamma_m\}$ be a support of β_0 and γ_0, respectively. Thus, for $i = 0, \ldots, n$ one has:

$$\begin{cases} \beta \equiv \beta_0 \\ \beta_i = \sum_{r=1}^{k} \sigma_i \beta_{ri} + \epsilon(\beta_i), \end{cases}$$

where β_{il}, for $i = 0, \ldots, n$ and $l = 1, \ldots, k$, is a linear combination of elements in $\pi(\beta_0)$.

For $j = 0, \ldots, m$:

$$\begin{cases} \gamma \equiv \gamma_0 \\ \gamma_j = \sum_{r=1}^{k} \sigma_i \gamma_{rj} + \epsilon(\gamma_j), \end{cases}$$

where γ_{jl}, for $j = 0, \ldots, n$ and $l = 1, \ldots, k$, is a linear combination of elements in $\pi(\gamma_0)$.

Consider $\alpha = \beta_0 \cap \gamma_0$, so

$$\pi(\beta_0 \cap \gamma_0) = \{\beta_{0i} \cap \gamma_{0j} \mid \beta_{0i} \in \pi(\beta_0), \gamma_{0j} \in \pi(\gamma_0)\}.$$

Since,

$$\begin{cases} \beta_0 = \sum_{i=1}^{k} \sigma_i \beta_{0i} + \epsilon(\beta_0) \\ \gamma_0 = \sum_{j=1}^{k} \sigma_i \gamma_{0j} + \epsilon(\gamma_0). \end{cases}$$

Therefore:

$$\begin{aligned} \beta_0 \cap \gamma_0 &= \sum_{i=1}^{k} \sigma_i \beta_{0i} + \epsilon(\beta_0) \cap \sum_{j=1}^{k} \sigma_i \beta_{0j} + \epsilon(\beta_0) \\ &= (\sigma_i \beta_{01} \cap \sigma_i \gamma_{01}) + \ldots + (\sigma_i \beta_{01} \cap \sigma_i \gamma_{0k}) + (\sigma_i \beta_{01} \cap \epsilon(\gamma_0)) + \ldots + \\ &\quad (\sigma_i \beta_{0k} \cap \sigma_i \gamma_{01}) + \ldots + (\sigma_i \beta_{0k} \cap \sigma_i \gamma_{0k}) + (\sigma_i \beta_{0k} \cap \epsilon(\gamma_0)) + \epsilon(\beta_0 \cap \gamma_0) \\ &= (\sigma_1 \cap \sigma_1)(\beta_{01} \cap \gamma_{01}) + \ldots + (\sigma_k \cap \sigma_k)(\beta_{0k} \cap \gamma_{0k}) + \epsilon(\beta_0 \cap \gamma_0) \\ &= (\sigma_1)(\beta_{01} \cap \gamma_{01}) + \ldots + (\sigma_k)(\beta_{0k} \cap \gamma_{0k}) + \epsilon(\beta_0 \cap \gamma_0) \\ &= \sum_{j=1}^{k} \sigma_i (\beta_{0i} \cap \gamma_{0i}) + \epsilon(\beta_0 \cap \gamma_0). \end{aligned}$$

Let $\pi(\beta_0) = \{\beta_1, \ldots, \beta_n\}$ be a suport of β_0. Thus, for $i = 0, \ldots, n$ one has

$$\begin{cases} \beta \equiv \beta_0 \\ \beta_i = \sum_{r=1}^{k} \sigma_i \beta_{ri} + \epsilon(\beta_i), \end{cases}$$

where β_{il}, for $i = 0, \ldots, n$ e $l = 1, \ldots, k$, is a linear combination of elements in $\pi(\beta_0)$.

$\square$

Example 2. *Consider the regular expression $\alpha = aa^* \cap a$ with $\Sigma = \{a\}$. The support for α is*

$$\pi(aa^* \cap aa) = \{\beta \cap \gamma \mid \beta \in \pi(aa^*), \gamma \in \pi(a)\}$$

Since,

$$\begin{aligned}\pi(aa^*) &= \{\beta a^* \mid \beta \in \pi(a)\} \cup \pi(a^*)\\ &= \{\beta a^* \mid \beta \in \{\epsilon\}\} \cup \{a^*\}\\ &= \{\epsilon\}\} \cup \{a^*\}\\ &= \{\epsilon\, a^*\}\end{aligned}$$

and

$$\pi(aa) = \{a, \epsilon\}.$$

Thus,

$$\begin{aligned}\pi(a^* \cap aa) &= \{\beta \cap \gamma \mid \beta \in \{\epsilon\, a^*\}, \gamma \in \{a, \epsilon\}\}\\ &= \{\epsilon \cap a, \epsilon \cap \epsilon, a^* \cap a, a * \cap \epsilon\}\\ &= \{\epsilon, a^* \cap a\}\end{aligned}$$

For the regular expression $\beta = a^* \cap a$, we have

$$\begin{aligned}\pi(a^* \cap a) &= \{\beta \cap \gamma \mid \beta \in \pi(a^*), \gamma \in \pi(a)\}\\ &= \{\beta \cap \gamma \mid \beta \in \{a^*\}, \gamma \in \{\epsilon\}\}\\ &= \{a^* \cap \epsilon\}.\end{aligned}$$

Note that in the second example of Example 2 the closure of partial derivatives of the regular expression $a^* \cap a$ is equal to support $\pi(a^* \cap a)$ of it.

3 XRE in FAdo

The XRE is a class for the extended regular expressions in FAdo system that preserves the modulo-aci properties in a way to assure the finitude of some algorithms, such as *dfaDerivatives* (construction of Derivative DFA [Brz64]), *nfaPD* (Partial Derivative nfa [Ant96])) and *equivP* (verifies if two regular expressions are equivalent [Brz64]).
The intersections and disjunctions were implemented in XRE as sets of regular expressions, because it guarantees the modulo-aci prooperties. Consider as an example the following correspondences:

$$\begin{aligned}a + b^*c + \emptyset + a + \epsilon &\to \{a, b^*c, \epsilon\}\\ a \cap a^*a \cap \emptyset \cap a \cap \epsilon &\to \{a, a^*a, \epsilon\}\end{aligned}$$

The concatenations were represented as ordered lists, which allows to take advantage of the associative concatenation, for example:

$$a(a + c)^*a \to [a, (a + c)^*, a]$$

It was used the object-oriented paradigma of programming for the implementation of regular expressions, it had been used a different class for each of the operators ($+$, . , $\cap$, $\neg$, $*$). Figure 1 presents the classes for XRE and the principal methods coded. The *xre* class is the base class for all extended regular expression and the subclasses xsigmaP and xsigmaS represent the regular expressions Σ^+ and Σ^*, respectively. The methods *derivative*, *partialDerivative* and *linearForm* are implemented for each subclass. The method *support*, that is defined and proved below, is not implemented for xnot. The same occurs for the method *nfaGlushkov* [Glu61] that is only for non-extended regular expressions. The algorithm *equivP* defined in [Alm11] verifies if two regulares expressions are equal by creating both derivative automaton. Since we extended the derivatives in FAdo for intersection and negation, the *equivP* algorithm was extended too. Likewise, the algorithm *PD* that creates the closure of the partial derivatives of an *extended* regular expression in relation to all symbols occuring in it.

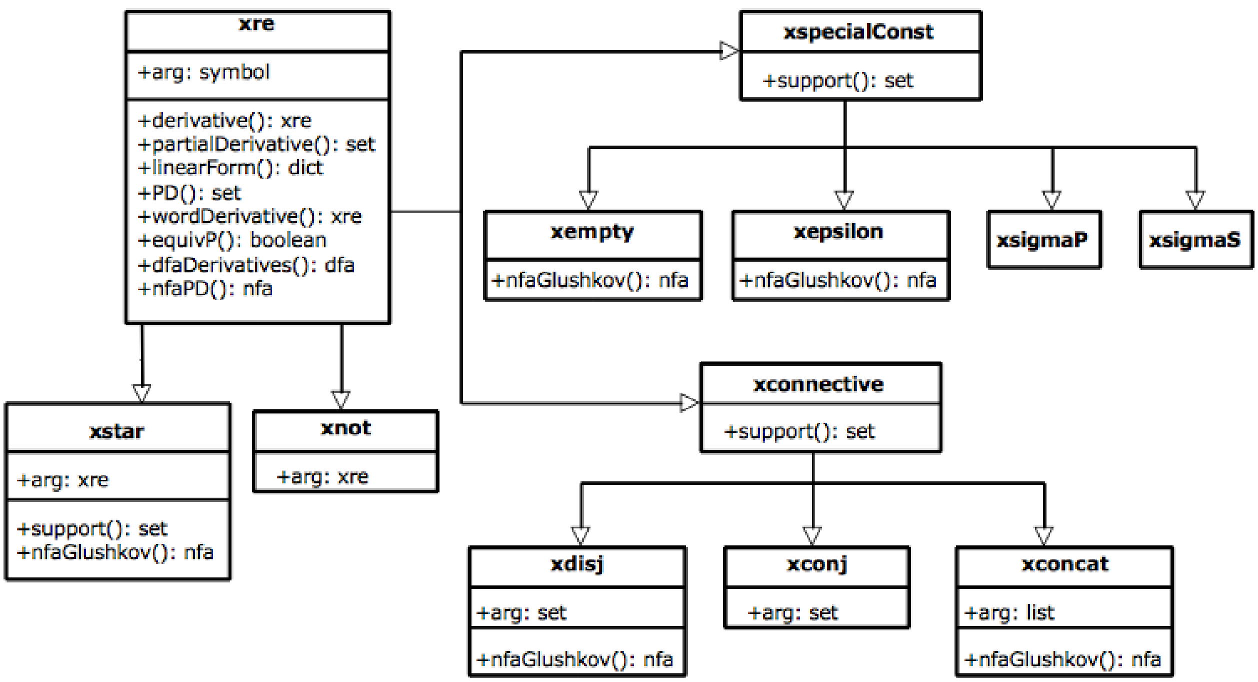

Figure 1: Classes for XRE.

References

[Alm11] Marco Almeida. *Equivalence of regular languages: an algorithmic approach and complexity analysis*. PhD thesis, University of Porto, April 2011.

[AM94] V. M. Antimirov and P. D. Mosses. Rewriting extended regular expressions. In G. Rozenberg and A. Salomaa, editors, *Developments in Language Theory*, pages 195 – 209. World Scientific, 1994.

[Ant96] V. M. Antimirov. Partial derivatives of regular expressions and finite automaton constructions. *Theoret. Comput. Sci.*, 155(2):291–319, 1996.

[Brz64] J. A. Brzozowski. Derivatives of regular expressions. *JACM*, 11(4):481–494, October 1964.

[CZ01] J. M. Champarnaud and D. Ziadi. From Mirkin's prebases to Antimirov's word partial derivatives. *Fundam. Inform.*, 45(3):195–205, 2001.

[Glu61] V. M. Glushkov. The abstract theory of automata. *Russian Math. Surveys*, 16:1–53, 1961.

[Mir66] B. G. Mirkin. An algorithm for constructing a base in a language of regular expressions. *Engineering Cybernetics*, 5:51—57, 1966.

[PCM11] J.-M. Champarnaud P. Caron and L. Mignot. Partial derivatives of an extended regular expression. 6638:179–19, 2011.

[Sal66] A. Salomaa. Two complete axiom systems for the algebra of regular events. *Journal of the Association for Computing Machinery*, 13(1):158–169, 1966.