On the Bisilimarity of the Position Automata

Eva Maia Nelma Moreira Rogério Reis
e-mail:{emaia,nam,rvr}@dcc.fc.up.pt
DCC-FC & CMUP, Universidade do Porto
Rua do Campo Alegre 1021, 4169-007 Porto, Portugal

Technical Report Series: DCC-2014-02

Departamento de 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/

On the Bisilimarity of the Position Automata

Eva Maia, Nelma Moreira, Rogério Reis
{emaia,nam,rvr}@dcc.fc.up.pt
DCC-FC & CMUP, Universidade do Porto

May 21, 2014

Abstract

Minimization of nondeterministic finite automata (NFA) is a hard problem (PSPACE-complete). Bisimulations are then an attractive alternative for reducing the size of NFAs, as even bisimilarity (the largest bisimulation) is almost linear using the Paige and Tarjan algorithm. NFAs obtained from regular expressions (REs) can have the number of states linear with respect to the size of the REs and conversion methods from REs to equivalent NFAs can produce NFAs without or with transitions labelled with the empty word (ε-NFA). The standard conversion without ε-transitions is the position automaton, $\mathcal{A}_{pos}$. Other conversions, such as partial derivative automata ($\mathcal{A}_{pd}$) or follow automata ($\mathcal{A}_f$), were proven to be quotients of the position automata (by some bisimulations). Recent experimental results suggested that for REs in (normalized) star normal form the position bisimilarity almost coincide with the $\mathcal{A}_{pd}$ automaton. Our goal is to have a better characterization of $\mathcal{A}_{pd}$ automata and their relation with the bisimilarity of the position automata. In this paper, we consider $\mathcal{A}_{pd}$ automata for regular expressions without Kleene star and establish under which conditions they are isomorphic to the bisimilarity of $\mathcal{A}_{pos}$.

1 Introduction

Regular expressions (REs), because of their succinctness and clear syntax, are the common choice to represent regular languages. The minimal deterministic finite automaton (DFA) equivalent to a RE can be exponentially larger than the RE. However, nondeterministic finite automata (NFAs) equivalent to REs can have the number of states linear with respect to (w.r.t) the size of the REs. But, minimization of NFAs is a hard problem (PSPACE-complete). Bisimulations are then an attractive alternative for reducing the size of NFAs, as even bisimilarity (the largest bisimulation) can be computed in almost linear time using the Paige and Tarjan algorithm [20].

Conversion methods from REs to equivalent NFAs can produce NFAs without or with transitions labelled with the empty word (ε-NFA). The standard conversion without ε-transitions is the position automaton ($\mathcal{A}_{pos}$) [12, 17]. Other conversions such as partial derivative automata ($\mathcal{A}_{pd}$) [1, 18], follow automata ($\mathcal{A}_f$) [14], or the construction by Garcia et al. ($\mathcal{A}_u$) [11] were proved to be quotients of the position automata, by specific bisimulations[1] [10, 14]. When REs are in (normalized) star normal form, i.e. when subexpressions of the star operator do not accept ε, the $\mathcal{A}_{pd}$ automaton is a quotient of the $\mathcal{A}_f$ [8].

[1] Also called right-invariant equivalence relations.

The $\mathcal{A}_{pos}$ bisimilarity was studied in [15], and of course it is always not larger than all other quotients. Nevertheless, some experimental results on uniform random generated REs suggested that for REs in (normalized) star normal form the $\mathcal{A}_{pos}$ bisimilarity automata almost coincide with the $\mathcal{A}_{pd}$ automata [13].

Our goal is to have a better characterization of $\mathcal{A}_{pd}$ automata and their relation with the $\mathcal{A}_{pos}$ bisimilarity. All the above mentioned automata ($\mathcal{A}_{pos}$, $\mathcal{A}_{pd}$, $\mathcal{A}_f$, and $\mathcal{A}_u$) can be obtained from a given RE by specific algorithms (without considering the correspondent bisimulation of $\mathcal{A}_{pos}$) in quadratic time. We aim to obtain a similar algorithm that computes, directly from a regular expression, the position bisimilarity automaton.

In this paper, we review the construction of $\mathcal{A}_{pd}$ as a quotient of $\mathcal{A}_{pos}$ and study several of its properties. For regular expressions without Kleene star we characterize the $\mathcal{A}_{pd}$ automata and we prove that the $\mathcal{A}_{pd}$ automaton is isomorphic to the position bisimilarity automaton, under certain conditions. Thus, for these special regular expressions, we conclude that the $\mathcal{A}_{pd}$ is an optimal conversion method. We close considering the difficulties of relating the two automata for general regular expressions.

2 Regular Expressions and Automata

Given an alphabet $\Sigma = \{\sigma_1, \sigma_2, \ldots, \sigma_k\}$ of size k, the set RE of *regular expressions* α over Σ is defined by the following grammar:

$$\alpha := \emptyset \mid \varepsilon \mid \sigma_1 \mid \cdots \mid \sigma_k \mid (\alpha + \alpha) \mid (\alpha \cdot \alpha) \mid (\alpha)^\star, \tag{1}$$

where the symbol $\cdot$ is often omitted. If two regular expressions α and β are syntactically equal, we write $\alpha \equiv \beta$. The *size* of a regular expression α, $|\alpha|$, is its number of symbols, disregarding parenthesis; its *alphabetic size*, $|\alpha|_\Sigma$, is the number of occurrences of letters from Σ; and $|\alpha|_\varepsilon$ denotes the number of occurrences of ε in α. A regular expression α is *linear* if all its letters are distinct.

The language represented by a RE α is denoted by $\mathcal{L}(\alpha)$. Two REs α and β are *equivalent* if $\mathcal{L}(\alpha) = \mathcal{L}(\beta)$, and one writes $\alpha = \beta$. We define $\varepsilon(\alpha) = \varepsilon$ if $\varepsilon \in \mathcal{L}(\alpha)$ and $\varepsilon(\alpha) = \emptyset$, otherwise. We can inductively define $\epsilon(\alpha)$ as follows:

$$\begin{aligned} \varepsilon(\sigma) = \varepsilon(\emptyset) &= \emptyset \\ \varepsilon(\varepsilon) &= \varepsilon \\ \varepsilon(\alpha^*) &= \varepsilon \end{aligned} \qquad \begin{aligned} \varepsilon(\alpha+\beta) &= \begin{cases} \varepsilon & \text{if } (\varepsilon(\alpha)=\varepsilon) \ \vee \ (\varepsilon(\beta)=\varepsilon) \\ \emptyset & \text{otherwise} \end{cases} \\ \varepsilon(\alpha\beta) &= \begin{cases} \varepsilon & \text{if } (\varepsilon(\alpha)=\varepsilon) \ \wedge \ (\varepsilon(\beta)=\varepsilon) \\ \emptyset & \text{otherwise} \end{cases} \end{aligned}$$

The algebraic structure $(\mathsf{RE}, +, ., \emptyset, \varepsilon)$ constitutes an idempotent semiring, and with the Kleene star operator $\star$, a Kleene algebra. The axioms for the star operator can be defined by the following rules [16]:

$$\varepsilon + \alpha\alpha^\star = \alpha^\star \text{ and } \varepsilon + \alpha^\star\alpha = \alpha^\star,$$
$$\beta + \alpha\gamma \leq \gamma \implies \alpha^\star\beta \leq \gamma \text{ and } \beta + \gamma\alpha \leq \gamma \implies \beta\alpha^\star \leq \gamma,$$

where $\alpha \leq \beta$ means $\alpha+\beta = \beta$. Given a language $L \subseteq \Sigma^\star$ and a word $w \in \Sigma^\star$, the *left-quotient* of L w.r.t. w is the language $w^{-1}L = \{x \mid wx \in L\}$. Brzozowski [6] defined the syntactic notion of derivative of a RE α w.r.t. a word w, $d_w(\alpha)$, such that $\mathcal{L}(d_w(\alpha)) = w^{-1}\mathcal{L}(\alpha)$, and showed that the set of derivatives of a regular expression w.r.t. all words is finite, modulo associativity (A), commutativity (C), and idempotence (I) of $+$ (which we denote by modulo ACI).

In this paper, we only consider REs α *normalized* under the following conditions:

- The expression α is *reduced* according to:
  - the equations $\emptyset + \alpha = \alpha + \emptyset = \alpha$, $\varepsilon.\alpha = \alpha.\varepsilon = \alpha$, $\emptyset.\alpha = \alpha.\emptyset = \emptyset$;
  - and the rule, for all subexpressions β of α, $\beta = \gamma + \varepsilon \implies \varepsilon(\gamma) = \emptyset$.
- The expression α is in *star normal form* (snf) [5], i.e. for all subexpressions $\beta^\star$ of α, $\varepsilon(\beta) = \emptyset$.

Every regular expression can be converted into an equivalent normalized RE in linear time.

A *nondeterministic finite automaton* (NFA) is a five-tuple $A = (Q, \Sigma, \delta, q_0, F)$ where Q is a finite set of states, Σ is a finite alphabet, q_0 in Q is the initial state, $F \subseteq Q$ is the set of final states, and $\delta : Q \times \Sigma \to 2^Q$ is the transition function. This transition function can be extended to words in the natural way. The language accepted by A is $\mathcal{L}(A) = \{w \in \Sigma^\star \mid \delta(q, w) \cap F \neq \emptyset\}$. Two NFAs are *equivalent* if they accept the same language. If two NFAs A and B are isomorphic, we write $A \simeq B$. An NFA is deterministic (DFA) if for all $(q, \sigma) \in Q \times \Sigma$, $|\delta(q, \sigma)| \leq 1$. A DFA is minimal if there is no equivalent DFA with fewer states. Minimal DFAs are unique up to isomorphism.

A binary symmetric and reflexive relation R on Q is a *bisimulation* if $\forall p, q \in Q$ and $\forall \sigma \in \Sigma$ if pRq then

- $p \in F$ if and only if $q \in F$;
- $\forall p' \in \delta(p, \sigma)\ \exists q' \in \delta(q, \sigma)$ such that $p'Rq'$.

The sets of bisimulations on Q are closed under finite union. The largest bisimulation, i.e., the union of all bisimulation relations on Q, is called *bisimilarity* ($\equiv_b$), and it is an equivalence relation. Bisimilarity can be computed in almost linear time using the Paige and Tarjan algorithm [20]. If R is a equivalence bisimulation on Q the *quotient automaton* ${}^{A}\!/\!{}_{R}$ can be constructed by ${}^{A}\!/\!{}_{R} = ({}^{Q}\!/\!{}_{R}, \Sigma, {}^{\delta}\!/\!{}_{R}, [q_0], {}^{F}\!/\!{}_{R})$, where $[q]$ is the equivalence class that contains $q \in Q$; ${}^{S}\!/\!{}_{R} = \{[q] \mid q \in S\}$, with $S \subseteq Q$; and ${}^{\delta}\!/\!{}_{R} = \{([p], \sigma, [q]) \mid (p, \sigma, q) \in \delta\}$. It is easy to see that $\mathcal{L}({}^{A}\!/\!{}_{R}) = \mathcal{L}(A)$. The quotient automaton ${}^{A}\!/\!{}_{\equiv_b}$ is the minimal automaton among all quotient automata ${}^{A}\!/\!{}_{R}$, where R is a bisimulation on Q, and it is unique up to isomorphism. By language abuse, we will call ${}^{A}\!/\!{}_{\equiv_b}$ the *bisimilarity* of automaton A. If A is a DFA, ${}^{A}\!/\!{}_{\equiv_b}$ is the minimal DFA equivalent to A.

2.1 Position Automaton

The position automaton was introduced independently by Glushkov [12] and McNaughton and Yamada [17]. The states in the position automaton, equivalent to a regular expression α, correspond to the positions of letters in α plus an additional initial state. Let $\overline{\alpha}$ denote the linear regular expression obtained by marking each letter with its position in α, i.e., $\mathcal{L}(\overline{\alpha}) \in \overline{\Sigma}^\star$ where $\overline{\Sigma} = \{\sigma_i \mid \sigma \in \Sigma, 1 \leq i \leq |\alpha|_\Sigma\}$. For example, the marked version of the regular expression $\tau = (ab^\star + b)^\star a$ is $\overline{\tau} = (a_1 b_2^\star + b_3)^\star a_4$. The same notation is used to remove the markings, i.e., $\overline{\overline{\alpha}} = \alpha$. Let $\mathsf{Pos}(\alpha) = \{1, 2, \ldots, |\alpha|_\Sigma\}$, and $\mathsf{Pos}_0(\alpha) = \mathsf{Pos}(\alpha) \cup \{0\}$.

We can define the following three sets, where $i, j \in \mathsf{Pos}(\alpha)$: $\mathsf{First}(\alpha) = \{i | a_i w \in \mathcal{L}(\overline{\alpha})\}$, $\mathsf{Last}(\alpha) = \{i | w a_i \in \mathcal{L}(\overline{\alpha}))\}$, $\mathsf{Follow}(\alpha, i) = \{j | u a_i a_j v \in \mathcal{L}(\overline{\alpha}))\}$. It is necessary to extend

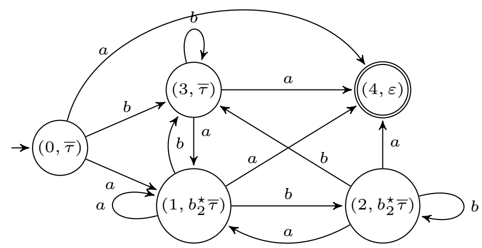


Figure 1: $\mathcal{A}_c(\tau)$.

$\mathsf{Follow}(\alpha, 0) = \mathsf{First}(\alpha)$ and define that $\mathsf{Last}_0(\alpha)$ is $\mathsf{Last}(\alpha)$ if $\varepsilon(\alpha) = \emptyset$, or $\mathsf{Last}(\alpha) \cup \{0\}$ otherwise.

The position automaton for α is

$$\mathcal{A}_{pos}(\alpha) = (\mathsf{Pos}_0(\alpha), A, \delta_{pos}, 0, \mathsf{Last}_0(\alpha))$$

where $\delta_{pos}(i, a) = \{j | j \in \mathsf{Follow}(\alpha, i), a = \overline{a_j}\}$. The position automata can be computed in quadratic time.

2.2 c-Continuation Automaton

Berry and Sethi [3] and Champarnaud and Ziadi [10] define the c-continuation automaton which is isomorphic to the $\mathcal{A}_{pos}$ and it is useful to obtains the $\mathcal{A}_{pd}$ automata in an efficient way.

If α is linear, for every symbol $\sigma \in \overline{\Sigma}$ and every word $w \in \overline{\Sigma}^\star$, $d_{w\sigma}(\alpha)$ is either $\emptyset$ or unique modulo ACI [3]. If $d_{w\sigma}(\alpha)$ is different from $\emptyset$, it is named *c-continuation* of α w.r.t. $\sigma \in \overline{\Sigma}$, denoted by $c_\sigma(\alpha)$ and it is defined as follows:

$$\begin{aligned} c_\sigma(\emptyset) &= c_\sigma(\varepsilon) = \emptyset \\ c_\sigma(\sigma') &= \begin{cases} \{\varepsilon\}, & \text{if } \sigma' = \sigma \\ \emptyset, & \text{otherwise} \end{cases} \\ c_\sigma(\alpha^\star) &= c_\sigma(\alpha)\alpha^\star \end{aligned} \qquad \begin{aligned} c_\sigma(\alpha + \beta) &= \begin{cases} c_\sigma(\alpha), & \text{if } c_\sigma(\alpha) \neq \emptyset \\ c_\sigma(\beta), & \text{otherwise} \end{cases} \\ c_\sigma(\alpha\beta) &= \begin{cases} c_\sigma(\alpha)\beta, & \text{if } c_\sigma(\alpha) \neq \emptyset \\ c_\sigma(\beta), & \text{otherwise} \end{cases} \end{aligned} \tag{2}$$

We also define $c_0(\alpha) = d_\varepsilon(\alpha) = \alpha$. This means that we can associate to each position $i \in \mathsf{Pos}_0(\alpha)$, a unique c-continuation. For example, given $\overline{\tau} = (a_1 b_2^\star + b_3)^\star a_4$ we have $c_{a_1}(\overline{\tau}) = b_2^\star\overline{\tau}$, $c_{b_2}(\overline{\tau}) = b_2^\star\overline{\tau}$, $c_{b_3}(\overline{\tau}) = \overline{\tau}$, and $c_{a_4}(\overline{\tau}) = \varepsilon$. The c-continuation automaton for α is $\mathcal{A}_c(\alpha) = (Q_c, \Sigma, \delta_c, q_0, F_c)$ where $Q_c = \{q_0\} \cup \{(i, c_{\sigma_i}(\overline{\alpha})) \mid i \in \mathsf{Pos}(\alpha)\}$, $q_0 = (0, c_0(\overline{\alpha}))$, $F_c = \{(i, c_{\sigma_i}(\overline{\alpha})) \mid \varepsilon(c_{\sigma_i}(\overline{\alpha})) = \varepsilon\}$, $\delta_c = \{((i, c_{\sigma_i}(\overline{\alpha})), b, (j, c_{\sigma_j}(\overline{\alpha}))) \mid \overline{\sigma_j} = b \wedge d_{\sigma_j}(c_{\sigma_i}(\overline{\alpha})) \neq \emptyset\}$. The $\mathcal{A}_c(\tau)$ is represented in Figure 1.

If we ignore the c-continuations in the label of each state, we obtain the position automaton.

Proposition 1 (Champarnaud & Ziadi)**.** *$\forall\alpha \in \mathsf{RE}$, $\mathcal{A}_{pos}(\alpha) \simeq \mathcal{A}_c(\alpha)$.*

We can establish a relation between the sets First, Follow and Last and the c-continuations:

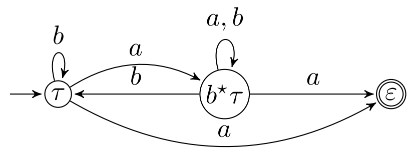


Figure 2: $\mathcal{A}_{pd}(\tau)$.

Proposition 2 (Champarnnaud & Ziadi). *For all* $\alpha \in \mathsf{RE}$*, the following equalities hold*

$$\begin{aligned} First(\alpha) &= \{\sigma \in \overline{\Sigma} | d_a(\overline{\alpha}) \neq \emptyset\} \\ Last(\alpha) &= \{\sigma \in \overline{\Sigma} | \varepsilon(c_\sigma(\overline{\alpha})) \neq \emptyset\} \\ Follow(\alpha, i) &= \{\sigma_j \in \overline{\Sigma} | d_{\sigma_j}(c_{\sigma_i}(\overline{\alpha})) \neq \emptyset\} \end{aligned}$$

2.3 Partial Derivative Automaton

The partial derivative automaton of a regular expression was introduced independently by Mirkin [18] and Antimirov [1]. Champarnaud and Ziadi [9] proved that the two formulations are equivalent. For a RE α and a symbol $\sigma \in \Sigma$, the set of partial derivatives of α w.r.t. σ is defined inductively as follows:

$$\begin{aligned} \partial_\sigma(\emptyset) &= \partial_\sigma(\varepsilon) = \emptyset & \partial_\sigma(\alpha + \beta) &= \partial_\sigma(\alpha) \cup \partial_\sigma(\beta) \\ \partial_\sigma(\sigma') &= \begin{cases} \{\varepsilon\}, & \text{if } \sigma' = \sigma \\ \emptyset, & \text{otherwise} \end{cases} & \begin{aligned} \partial_\sigma(\alpha\beta) \\ \partial_\sigma(\alpha^\star) \end{aligned} & \begin{aligned} &= \partial_\sigma(\alpha)\beta \cup \varepsilon(\alpha)\partial_\sigma(\beta) \\ &= \partial_\sigma(\alpha)\alpha^\star \end{aligned} \end{aligned} \tag{3}$$

where for any $S \subseteq \mathsf{RE}$, $\beta \in \mathsf{RE}$, $S\emptyset = \emptyset S = \emptyset$, $S\varepsilon = \varepsilon S = S$, and $S\beta = \{\alpha\beta | \alpha \in S\}$ if $\beta \neq \emptyset$, and $\beta \neq \varepsilon$.

The definition of partial derivative can be extended to sets of regular expressions, words, and languages. Given $\alpha \in \mathsf{RE}$ and $\sigma \in \Sigma$, $\partial_\sigma(S) = \bigcup_{\alpha \in S} \partial_\sigma(\alpha)$ for $S \subseteq \mathsf{RE}$, $\partial_\varepsilon(\alpha) = \alpha$ and $\partial_{w\sigma}(\alpha) = \partial_\sigma(\partial_w(\alpha))$, for any $w \in \Sigma^\star, \sigma \in \Sigma$, and $\partial_L(\alpha) = \bigcup_{w \in L} \partial_w(\alpha)$ for $L \subseteq \Sigma^\star$. We know that $\bigcup_{\tau \in \partial_w(\alpha)} \mathcal{L}(\tau) = w^{-1}\mathcal{L}(\alpha)$. The set of all partial derivatives of α w.r.t. words is denoted by $\mathsf{PD}(\alpha) = \bigcup_{w \in \Sigma^\star} \partial_w(\alpha)$. Note that the set $\mathsf{PD}(\alpha)$ is always finite [1], as opposed to what happens for the Brzozowski derivatives set which is only finite modulo ACI.

The partial derivative automaton is defined by $\mathcal{A}_{pd}(\alpha) = (\mathsf{PD}(\alpha), \Sigma, \delta_{pd}, \alpha, F_{pd})$, where $\delta_{pd} = \{(\tau, \sigma, \tau') \mid \tau \in \mathsf{PD}(\alpha) \text{ and } \tau' \in \partial_\sigma(\tau)\}$ and $F_{pd} = \{\tau \in \mathsf{PD}(\alpha) \mid \varepsilon(\tau) = \varepsilon\}$. Considering $\tau = (ab^\star + b)^\star a$, the Figure 2 shows $\mathcal{A}_{pd}(\tau)$.

Note that if α is a linear regular expression, for every word w, $|\partial_w(\alpha)| \leq 1$ and the partial derivative coincide with $d_w(\alpha)$ modulo ACI. Given the c-continuation automaton $\mathcal{A}_c(\alpha)$, let $\equiv_c$ be the bisimulation on Q_c defined by $(i, c_{\sigma_i}(\overline{\alpha})) \equiv_c (j, c_{\sigma_j}(\overline{\alpha}))$ if $\overline{c_{\sigma_i}(\overline{\alpha})} \equiv \overline{c_{\sigma_j}(\overline{\alpha})}$. That the $\mathcal{A}_{pd}$ is isomorphic to the resulting quotient automaton, follows from the proposition below. For our running example, we have $(0, c_\varepsilon) \equiv_c (3, c_{b_3})$ and $(1, c_{a_1}) \equiv_c (2, c_{b_2})$. In Figure 2, we can see the merged states, and that the corresponding REs are unmarked.

Proposition 3 (Champarnaud & Ziadi). $\forall \alpha \in \mathsf{RE}$*,* $\mathcal{A}_{pd}(\alpha) \simeq {\mathcal{A}_c(\alpha)}/{\equiv_c}$*.*

2.3.1 Inductive Characterization of $\mathcal{A}_{pd}$

Mirkin's construction of the $\mathcal{A}_{pd}(\alpha)$ is based on solving a system of equations $\alpha_i = \sigma_1\alpha_{i1} + \ldots + \sigma_k\alpha_{ik} + \varepsilon(\alpha_i)$, with $\alpha_0 \equiv \alpha$ and α_{ij}, $1 \leq j \leq k$, linear combinations the α_i, $0 \leq i \leq n$, $n \geq 0$. A solution $\pi(\alpha) = \{\alpha_1, \ldots, \alpha_n\}$ can be obtained inductively on the structure of α as follows:

$$\begin{aligned} \pi(\emptyset) &= \emptyset & \qquad \pi(\alpha \cup \beta) &= \pi(\alpha) \cup \pi(\beta) \\ \pi(\varepsilon) &= \emptyset & \pi(\alpha\beta) &= \pi(\alpha)\beta \cup \pi(\beta) \\ \pi(\sigma) &= \{\varepsilon\} & \pi(\alpha^\star) &= \pi(\alpha)\alpha^\star. \end{aligned} \tag{4}$$

Champarnaud and Ziadi [9] proved that $\mathsf{PD}(\alpha) = \pi(\alpha) \cup \{\alpha\}$ and that the two constructions led to the same automaton.

As noted by Broda et. al [4], Mirkin's algorithm to compute $\pi(\alpha)$ also provides an inductive definition of the set of transitions of $\mathcal{A}_{pd}(\alpha)$. Let $\varphi(\alpha) = \{(\sigma, \gamma) \mid \gamma \in \partial_\sigma(\alpha), \sigma \in \Sigma\}$ and $\lambda(\alpha) = \{\alpha' \mid \alpha' \in \pi(\alpha),\ \varepsilon(\alpha') = \varepsilon\}$, where both sets can be inductively defined using (3) and (4). We have, $\delta_{pd} = \{\alpha\} \times \varphi(\alpha) \cup F(\alpha)$ where the result of the $\times$ operation is seen as a set of triples and the set F is defined inductively by:

$$\begin{aligned} F(\emptyset) &= F(\varepsilon) = F(\sigma) = \emptyset,\ \sigma \in \Sigma \\ F(\alpha + \beta) &= F(\alpha) \cup F(\beta) \\ F(\alpha\beta) &= F(\alpha)\beta \cup F(\beta) \cup \lambda(\alpha)\beta \times \varphi(\beta) \\ F(\alpha^\star) &= F(\alpha)\alpha^\star \cup (\lambda(\alpha) \times \varphi(\alpha))\alpha^\star. \end{aligned} \tag{5}$$

Then, we can inductively construct the partial derivative automaton of α using the following result.

Proposition 4. *For all $\alpha \in \mathsf{RE}$, and $\lambda'(\alpha) = \lambda(\alpha) \cup \varepsilon(\alpha)\{\alpha\}$,*

$$\mathcal{A}_{pd}(\alpha) = (\pi(\alpha) \cup \{\alpha\}, \Sigma, \{\alpha\} \times \varphi(\alpha) \cup F(\alpha), \alpha, \lambda'(\alpha)),$$

Proof. Note that the sets F, λ and φ correspond, respectively, to the sets Follow, Last, and First, modulo the equivalence relation that defines $\mathcal{A}_{pd}$ as a quotient of $\mathcal{A}_{pos}$. We can define inductively $\mathcal{A}_{pd}(\alpha)$ on the structure of α. Thus if α is ε, $\pi(\alpha) = \emptyset$ and $\delta_{pd} = \{\varepsilon\} \times \varphi(\alpha) \cup F(\alpha) = \emptyset$, where $\varphi(\alpha) = \{(\sigma, \gamma) \mid \gamma \in \partial_\sigma(\alpha), \sigma \in \Sigma\} = \emptyset$. Therefore, $\mathcal{A}_{pd}(\varepsilon) = (\{\varepsilon\}, \emptyset, \emptyset, \varepsilon, \{\varepsilon\})$. If α is $\emptyset$, it is easy to see that $\mathcal{A}_{pd}(\emptyset) = (\{\emptyset\}, \emptyset, \emptyset, \emptyset, \emptyset)$.
If α is σ, then $\pi(\sigma) = \varepsilon$ and $\delta_{pd} = \{\sigma\} \times \varphi(\alpha) \cup F(\alpha) = \{(\sigma, \sigma, \varepsilon)\}$, because $\varphi(\alpha) = \{(\sigma, \gamma) \mid \gamma \in \partial_\sigma(\alpha), \sigma \in \Sigma\} = \{(\sigma, \varepsilon)\}$. Therefore, $\mathcal{A}_{pd}(\sigma) = (\{\sigma, \varepsilon\}, \{\sigma\}, \{(\sigma, \sigma, \varepsilon)\}, \sigma, \{\varepsilon\})$.
If α is $\gamma + \beta$ then $\pi(\gamma + \beta) = \pi(\gamma) \cup \pi(\beta)$ and

$$\begin{aligned} \delta_{pd} &= \{\gamma + \beta\} \times \varphi(\gamma + \beta) \cup F(\gamma + \beta) \\ &= \{\gamma + \beta\} \times \varphi(\gamma) \cup \{\gamma + \beta\} \times \varphi(\beta) \cup F(\gamma) \cup F(\beta), \end{aligned}$$

because $\varphi(\gamma + \beta) = \{(\sigma, \theta) \mid \theta \in \partial_\sigma(\gamma), \sigma \in \Sigma\} \cup \{(\sigma, \theta) \mid \theta \in \partial_\sigma(\beta), \sigma \in \Sigma\}$. Thus

$$\begin{aligned} \mathcal{A}_{pd}(\gamma + \beta) =& (\{\gamma + \beta\} \cup \pi(\gamma) \cup \pi(\beta), \Sigma, \{\gamma + \beta\} \times \varphi(\gamma) \cup \{\gamma + \beta\} \times \varphi(\beta) \cup F(\gamma) \cup F(\beta), \\ & \gamma + \beta, \lambda(\gamma) \cup \lambda(\beta) \cup \varepsilon(\gamma + \beta)\{\gamma + \beta\}). \end{aligned}$$

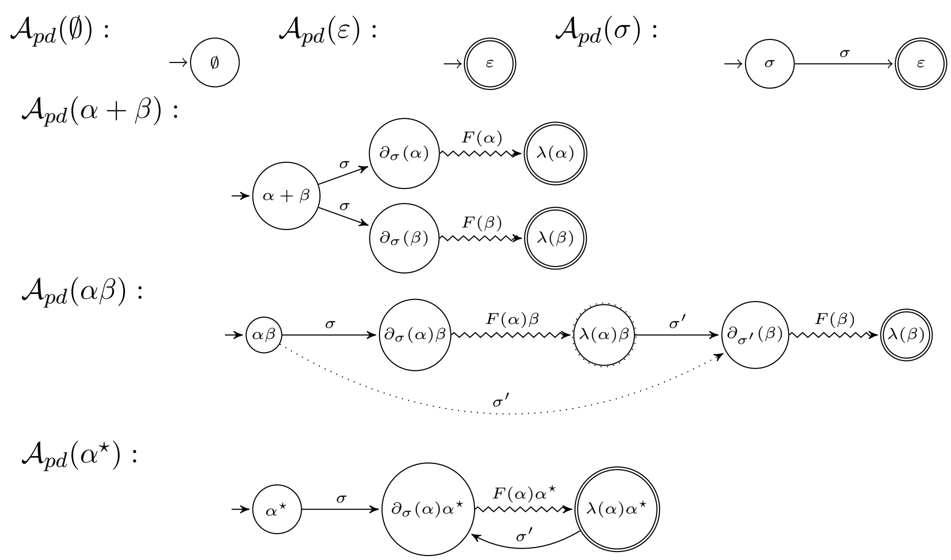


Figure 3: Inductive construction of $\mathcal{A}_{pd}$. The initial states are final if ε belongs to its language. Note that only if $\varepsilon(\beta)=\varepsilon$ the dotted arrow in $\mathcal{A}_{pd}(\alpha\beta)$ exists and the state $\lambda(\alpha)\beta$ is final.

If α is $\gamma\beta$, then $\pi(\gamma\beta)=\pi(\gamma)\beta\cup\pi(\beta)$ and

$$\begin{aligned}\delta_{pd} &= \{\gamma\beta\}\times\varphi(\gamma\beta)\cup F(\gamma\beta)\\ &= \{\gamma\beta\}\times\varphi(\gamma)\beta\cup\varepsilon(\gamma)(\{\gamma\beta\}\times\varphi(\beta))\cup F(\gamma)\beta\cup F(\beta)\cup\lambda(\gamma)\beta\times\varphi(\beta)\end{aligned}$$

because, $\varphi(\gamma\beta)=\{(\sigma,\theta\beta)\mid\theta\in\partial_\sigma(\gamma),\sigma\in\Sigma\}\cup\varepsilon(\gamma)(\{(\sigma,\theta)\mid\theta\in\partial_\sigma(\beta),\sigma\in\Sigma\})$. Thus

$$\begin{aligned}\mathcal{A}_{pd}(\gamma\beta) = (&\{\gamma\beta\}\cup\pi(\gamma)\beta\cup\pi(\beta),\Sigma,\\ &\{\gamma\beta\}\times\varphi(\gamma)\beta\cup\varepsilon(\gamma)(\{\gamma\beta\}\times\varphi(\beta))\cup F(\gamma)\beta\cup F(\beta)\cup\lambda(\gamma)\beta\times\varphi(\beta),\\ &\gamma\beta,\lambda(\beta)\cup\varepsilon(\beta)\lambda(\gamma)\beta\cup\varepsilon(\gamma\beta)\{\gamma\beta\}).\end{aligned}$$

If $\alpha=\beta^\star$ then $\pi(\beta^\star)=\pi(\beta)\beta^\star$ and $\delta_{pd}=\{\beta^\star\}\times\varphi(\beta)\beta^\star\cup F(\beta)\beta^\star\cup(\lambda(\beta)\times\varphi(\beta))\beta^\star$ because $\varphi(\beta^\star)=\{(\sigma,\theta\beta^\star)\mid\theta\in\partial_\sigma(\beta),\sigma\in\Sigma\}$. Thus,

$$\mathcal{A}_{pd}(\beta^\star)=(\{\beta^\star\}\cup\pi(\beta)\beta^\star,\Sigma,\{\beta^\star\}\times\varphi(\beta)\beta^\star\cup F(\beta)\beta^\star\cup(\lambda(\beta)\times\varphi(\beta))\beta^\star,\beta^\star,\lambda(\beta)\beta^\star).$$

$\square$

Figure 3 illustrates this inductive construction, where we assume that states are merged whenever they correspond to syntactically equal REs.

A new proof of Proposition 3 can also be given using the function π. Let π' be a function that coincides with π except that $\pi'(\sigma)=\{(\sigma,\varepsilon)\}$ and in the two last rules the regular expression, either β or $\alpha^\star$, is concatenated to the second component of each pair in π', i.e.,

$$\begin{aligned}\pi'(\emptyset) &= \emptyset & \pi'(\alpha\cup\beta) &= \pi'(\alpha)\cup\pi'(\beta)\\ \pi'(\varepsilon) &= \emptyset & \pi'(\alpha\beta) &= \pi'(\alpha)\beta\cup\pi'(\beta)\\ \pi'(\sigma) &= \{(\sigma,\varepsilon)\} & \pi'(\alpha^\star) &= \pi'(\alpha)\alpha^\star.\end{aligned} \tag{6}$$

Proposition 5. *Let $\alpha \in RE$, $\pi'(\overline{\alpha}) = \{(i, c_{\sigma_i}(\overline{\alpha}))|i \in \mathsf{Pos}(\overline{\alpha})\}$.*

Proof. First of all note that $\overline{\alpha+\beta} = \overline{\alpha} + \overline{\beta}$, $\overline{\alpha\beta} = \overline{\alpha}\overline{\beta}$, and $\overline{\alpha^\star} = \overline{\alpha}^\star$. Let us prove by induction on α. For the base cases it is easy to prove that the proposition holds. Suppose that the proposition holds for γ and β. If $\overline{\alpha}$ is $\overline{\gamma+\beta}$, then

$$\begin{aligned}
\pi'(\overline{\alpha}) &= \{(i, c_{\sigma_i}(\overline{\alpha})) \mid i \in \mathsf{Pos}(\overline{\alpha})\} \\
&= \{(i, c_{\sigma_i}(\overline{\gamma+\beta})) \mid i \in \mathsf{Pos}(\overline{\gamma+\beta})\} \\
&= \{(i, c_{\sigma_i}(\overline{\gamma}+\overline{\beta})) \mid i \in \mathsf{Pos}(\overline{\gamma}+\overline{\beta})\} \\
&= \{(i, c_{\sigma_i}(\overline{\gamma})) \mid i \in \mathsf{Pos}(\overline{\gamma}))\} \cup \{(i, c_{\sigma_i}(\overline{\beta})) \mid i \in \mathsf{Pos}(\overline{\beta})\} \text{ by the rules in (2)} \\
&= \pi'(\overline{\gamma}+\overline{\beta}) \;=\; \pi'(\overline{\gamma+\beta})
\end{aligned}$$

If $\overline{\alpha}$ is $\overline{\gamma\beta}$, then

$$\begin{aligned}
\pi'(\overline{\alpha}) &= \{(i, c_{\sigma_i}(\overline{\alpha})) \mid i \in \mathsf{Pos}(\overline{\alpha})\} \\
&= \{(i, c_{\sigma_i}(\overline{\gamma\beta})) \mid i \in \mathsf{Pos}(\overline{\gamma\beta})\} \\
&= \{(i, c_{\sigma_i}(\overline{\gamma}\overline{\beta})) \mid i \in \mathsf{Pos}(\overline{\gamma}\overline{\beta})\} \\
&= \{(i, c_{\sigma_i}(\overline{\gamma})\overline{\beta}) \mid i \in \mathsf{Pos}(\overline{\gamma})\} \cup \{(i, c_{\sigma_i}(\overline{\beta})) \mid i \in \mathsf{Pos}(\overline{\beta})\} \text{ by the rules in (2)} \\
&= \{(i, c_{\sigma_i}(\overline{\gamma})) \mid i \in \mathsf{Pos}(\overline{\gamma})\}\overline{\beta} \;\cup\; \{(i, c_{\sigma_i}(\overline{\beta})) \mid i \in \mathsf{Pos}(\overline{\beta})\} \\
&= \pi'(\overline{\gamma})\overline{\beta} \;\cup\; \pi'(\overline{\beta}) \;=\; \pi'(\overline{\gamma}\overline{\beta}) \;=\; \pi'(\overline{\gamma\beta})
\end{aligned}$$

If $\overline{\alpha}$ is $\overline{\gamma}^\star$, then

$$\begin{aligned}
\pi'(\overline{\alpha}) &= \{(i, c_{\sigma_i}(\overline{\alpha})) \mid i \in \mathsf{Pos}(\overline{\alpha})\} \\
&= \{(i, c_{\sigma_i}(\overline{\gamma}^\star)) \mid i \in \mathsf{Pos}(\overline{\gamma}^\star)\} \\
&= \{(i, c_{\sigma_i}(\overline{\gamma})\overline{\gamma}^\star) \mid i \in \mathsf{Pos}(\overline{\gamma}^\star)\} \text{ by the rules in (2)} \\
&= \{(i, c_{\sigma_i}(\overline{\gamma})) \mid i \in \mathsf{Pos}(\overline{\gamma}^\star)\}\overline{\gamma}^\star \\
&= \pi'(\overline{\gamma})\overline{\gamma}^\star \;=\; \pi'(\overline{\gamma}^\star)
\end{aligned}$$

Thus, the proposition holds. □

By Proposition 5, we can conclude that if we compute $\pi'(\overline{\alpha})$ we obtain exactly[2] the set of states $Q_c \setminus \{(0, c_\varepsilon)\}$ of the c-continuation automaton $\mathcal{A}_c(\alpha)$. Then it is easy to see that $\pi(\alpha)$ is obtained by unmarking the c-continuations and removing the first component of each pair, and thus ${Q_c}/{\equiv_c} = \pi(\alpha) \cup \{\alpha\}$. Considering $\overline{\tau} = (a_1 b_2^\star + b_3)^\star a_4$, $\pi'(\overline{\tau}) = \{(a_1, b_2^\star\overline{\tau}), (b_2, b_2^\star\overline{\tau}), (b_3, \overline{\tau}), (b_4, \varepsilon)\}$, which corresponds exactly to the set of states (excluding the initial) of $\mathcal{A}_c(\overline{\tau})$, presented in Figure 1. The set $\pi(\tau)$ is $\{b^\star\tau, \tau, \varepsilon\}$. That the other components are quotients, also follows.

2.4 Follow Automaton

In [14], Ilie and Yu proposed a new method to construct NFAs from regular expressions. First, the authors construct and NFA with ε-transitions – $A_f^\varepsilon(\alpha)$. Then they use an ε-elimination method to build the follow automaton – $A_f(\alpha)$. The authors also proved that the follow automaton is a quotient of the position automaton.

Proposition 6 (Ilie & Yu)**.** *For all $\alpha \in \mathsf{RE}$, $A_f(\alpha) \simeq {\mathcal{A}_{pos}(\alpha)}/{\equiv_f}$, where $i \equiv_f j$ iff both i, j or none belong to $last(\alpha)$ and $follow(\alpha, i) = follow(\alpha, j)$.*

[2] Considering, for each position i, the marked letter σ_i.

2.5 Garcia *et.al* Automaton

In [11], the authors also proposed a new method to construct NFAs from regular expressions. The resulting automaton size is bounded above by the size of the smallest automata obtained by the follow and partial derivatives methods.

Let the equivalence $\equiv_\vee$ be the join of the relations $\equiv_c$ and $\equiv_f$, where the join relation between two equivalence relations E_1 and E_2 is the smallest equivalence relation that contains E_1 and E_2. The Garcia et. al automaton is a quotient of the position automaton – $\mathcal{A}_u(\alpha) \simeq \mathcal{A}_{pos}(\alpha)/_{\vee}$.

3 $\mathcal{A}_{pd}$ Characterizations and Bisimilarity

We aim to obtain some characterizations of $\mathcal{A}_{pd}$ automaton and to determine when it coincides with the bisimilarity of the position automaton, i.e. $\mathcal{A}_{pos}/_{\equiv_b}$. We assume that all regular expressions are normalized. This ensures that the $\mathcal{A}_{pd}$ is a quotient of $\mathcal{A}_f$, so the smaller known direct ε-free automaton construction from a regular expression. As we discuss in Subsection 3.4, to solve the problem in the general case it is difficult, mainly because the lack of unique normal forms. Here, we give some partial solutions. First, we consider linear regular expressions and, in Subsection 3.2, we solve the problem for regular expressions representing finite languages.

3.1 Linear Regular Expressions

Given a linear regular expression α, it is obvious that the position automaton $\mathcal{A}_{pos}(\alpha)$ is a DFA. In this case, all positions correspond to distinct letters and transitions from a same state are all distinct. Thus, $\mathcal{A}_{pd}(\alpha)$ is also a DFA.

The following result is proved by Champarnaud and Ziadi in [8].

Proposition 7. *Let x and y be two positions of a normalized regular expression α. Then the following equivalence holds:*

$$c_{\sigma_x}(\alpha) \equiv c_{\sigma_y}(\alpha) \Leftrightarrow \forall a \in \mathsf{Pos}_0(\alpha) d_a(c_{\sigma_x}(\alpha)) \equiv d_a(c_{\sigma_y}(\alpha))$$

Proof. ($\Rightarrow$) It is obvious.
($\Leftarrow$) Let us suppose that $c_{\sigma_x}(\alpha) \not\equiv c_{\sigma_y}(\alpha)$. As $x \neq y$, there exists a subexpression of E, $E_x \bigotimes E_y$, with $\bigotimes \in \{+, .\}$ such that $x \in Pos_E(E_x)$ and $y \in Pos_E(E_y)$. If we look to the syntactic tree of E is not difficult to see that the reason for $c_{\sigma_x}(E) \not\equiv c_{\sigma_y}(E)$ is $c_{\sigma_x}(E_x) \not\equiv c_{\sigma_y}(E_y)$. Note that $\forall z \in E_x, z \notin E_y$ because E is marked. Thus, we have $c_{\sigma_x}(E) = A_1 \dots A_\alpha C_1 \dots C_m$ and $c_{\sigma_y}(E) = B_1 \dots B_\beta C_1 \dots C_m$. Note that A_i, B_i and C_i are subexpressions of E. By hypothesis we know that:

$$\begin{aligned} & d_{\sigma_z}(c_{\sigma_x}(E)) = d_{\sigma_z}(c_{\sigma_y}(E)) \\ \Leftrightarrow \quad & d_{\sigma_z}(A_1 \dots A_\alpha C_1 \dots C_m) = d_{\sigma_z}(B_1 \dots B_\beta C_1 \dots C_m) \end{aligned}$$

If $\sigma_z \notin A_i$ and $z \notin B_i$ it is easy to see that the equality holds. If $\sigma_z \in A_i$ and $\sigma_z \notin B_i$ then

$$d_{\sigma_z}(c_{\sigma_x}(E)) = d_{\sigma_z}(A_1 \dots A_\alpha C_1 \dots C_m)$$

and

$$\begin{array}{rcl} d_{\sigma_z}(c_{\sigma_y}(E)) & = & d_{\sigma_z}(B_1 \dots B_\beta C_1 \dots C_m) \\ & = & d_{\sigma_z}(C_1 \dots C_m) \qquad \lambda(B_1 \dots B_\beta) = 1 \end{array}$$

because $\sigma_z \notin B_i$. Note that if $\lambda(B_1 \dots B_\beta) \neq 1$ the equality in the hypothesis does not hold. Thus we have

$$d_{\sigma_z}(A_1 \dots A_\alpha C_1 \dots C_m) = d_{\sigma_z}(C_1 \dots C_m) \tag{7}$$

Suppose that $d_{\sigma_z}(A_1) \neq \emptyset$. Then $d_{\sigma_z}(A_1 \dots A_\alpha C_1 \dots C_m) \neq \emptyset$. And by the equality 7 we know that $d_{\sigma_z}(C_1 \dots C_m) \neq \emptyset$. But this means that we have $\sigma_z \in A_1 \dots A_\alpha$ and $\sigma_z \in C_1 \dots C_m$:

$$c_{\sigma_x}(E) = \underbrace{A_1 \dots A_\alpha}_{\sigma_z \in} \underbrace{C_1 \dots C_m}_{\sigma_z \in}$$

Thus, there exists k, $1 \leq k \leq m$ such that $C_k = F^\star$, and by the definition of d we also conclude that $\lambda(C_1 \dots C_k) = 1$. As $\sigma_z \in F$ we also conclude that $d_{\sigma_z}(F) \neq \emptyset$. As $\sigma_y \in E_y$, we know that $d_{\sigma_y}(E_y) \neq \emptyset$. We also know that $E_y \subseteq F^\star$ and $\lambda(C_1 \dots C_k) = 1$. Thus, we conclude that $d_{\sigma_y}(F) \neq \emptyset$. By the hypothesis we know that $d_{\sigma_y}(c_{\sigma_x}(E)) \neq \emptyset$. As $\sigma_y \notin E_x$ and $d_{\sigma_y}(A_1 \dots A_\alpha C_1 \dots C_m) \neq \emptyset$ we conclude that $\lambda(A_1 \dots A_\alpha C_1 \dots F^\star) = 1$. But $c_{\sigma_x}(F)$ is in $A_1 \dots A_\alpha C_1 \dots F^\star$, because $c_{\sigma_x}(F^\star) = c_{\sigma_x}(F)F^\star$. Thus $\lambda(c_{\sigma_x}(F)) = 1$. Since we have A_1 in $c_{\sigma_x}(E)$, there exists a subexpression in E with one of these two forms:

- $S_x A_1$, and in this case $c_{\sigma_x}(E) = c_{\sigma_x}(S_x)A_1$,
- $S_x^\star$, $c_{\sigma_x}(E) = c_{\sigma_x}(S_x^\star) = c_{\sigma_x}(S_x)S_x^\star$ and $A_1 = S_x^\star$,

such that $\sigma_x \in S_x$, S_x contains no occurrence of "." or "$\star$" and $c_{\sigma_x}(S_x) = \varepsilon$. Thus S_x is equal to σ_x or $\sigma_x + \gamma$ or $\gamma + \sigma_x$. Thus $\lambda(c_{\sigma_x}(S_x)) = 1$. We know that S_x is subexpression of $F^\star$. Thus,

$$d_{\sigma_z}(c_{\sigma_x}(F^\star)) = d_{\sigma_z}(F)F^\star = \dots = d_{\sigma_z}(A_1) \neq \emptyset.$$

Therefore exists an σ_z and σ_x such that $\lambda(c_{\sigma_x}(F)) = 1$, $d_{\sigma_z}(c_{\sigma_x}(F)) \neq \emptyset$ and $d_{\sigma_z}(F) \neq \emptyset$, which is a contradiction with E is in SNF. $\square$

Proposition 8. *If α is a normalized linear regular expression, $\mathcal{A}_{pd}(\alpha)$ is minimal.*

Proof. By [8, Theorem 2] we know that

$$c_{\sigma_x}(\alpha) \not\equiv c_{\sigma_y}(\alpha) \Leftrightarrow \{\sigma \mid \partial_\sigma(c_{\sigma_x}(\alpha)) \neq \emptyset\} \neq \{\sigma \mid \partial_\sigma(c_{\sigma_y}(\alpha)) \neq \emptyset\}$$

where α is a normalized linear regular expression and σ_x and σ_y are two distinct letters. We want to prove that any two states $c_{\sigma_x}(\alpha)$ and $c_{\sigma_y}(\alpha)$ of $\mathcal{A}_{pd}(\alpha)$ are distinguishable. Consider $\sigma' \in \Sigma$ such that $\sigma' \in \{\sigma \mid \partial_\sigma(c_{\sigma_x}(\alpha)) \neq \emptyset\}$ but $\sigma' \notin \{\sigma \mid \partial_\sigma(c_{\sigma_y}(\alpha)) \neq \emptyset\}$. Then $\delta_{pd}(c_{\sigma_x}(\alpha), \sigma') = c_{\sigma'}(\alpha)$. By construction, we know that $\exists w \in \Sigma^\star$ such that $\delta_{pd}(c_{\sigma'}(\alpha), w) \in F_{pd}$. Let $w' = \sigma' w$. Therefore $\delta_{pd}(c_{\sigma_x}(\alpha), w') = \delta_{pd}(c_{\sigma'}(\alpha), w) \in F_{pd}$ and either δ_{pd} is not defined for $(c_{\sigma_y}(\alpha), w')$ or $\delta_{pd}(c_{\sigma_y}(\alpha), w')$ is a non final dead state. Thus, the two states are distinguishable. $\square$

It follows, from this, that for any linear regular expressions α,

$$\mathcal{A}_{pd}(\alpha) \simeq \mathcal{A}_{pos}(\alpha) /_{\equiv_b}.$$

The two following results are proved by Champarnaud and Ziadi in [10].

Proposition 9. *Let α be a regular expression and α' a subexpression of α. For all α', $\sigma_i \in \overline{\Sigma}$ and $\sigma \in \Sigma$,*

$$\bigcup_{\overline{\sigma_i}=\sigma} \overline{d_{\sigma_i}(\overline{\alpha})} = \partial_\sigma(\alpha)$$

Proposition 10. *The relation $\equiv_c$ is right invariant.*

Proof. Let us consider the following equivalence:

$$(x, c_{\sigma_x}(\alpha)) \sim (y, c_{\sigma_y}(\alpha)) \Leftrightarrow x \equiv_c y \Leftrightarrow \overline{c_{\sigma_x}(\alpha)} \equiv \overline{c_{\sigma_y}(\alpha)}$$

We want to prove that
$\forall a \in \Sigma (x, c_{\sigma_x}(\alpha)) \sim (y, c_{\sigma_y}(\alpha)) \Rightarrow$

$$\begin{cases} (1) \quad \forall (z, c_{\sigma_z}(\alpha)) \in \delta((x, c_{\sigma_x}(\alpha)), a) \exists (w, c_{\sigma_w}(\alpha)) \in \delta((y, c_{\sigma_y}(\alpha)), a) \text{ such that } z \sim w \\ (2) \quad \forall (w, c_{\sigma_w}(\alpha)) \in \delta((y, c_{\sigma_y}(\alpha)), a) \exists (z, c_{\sigma_z}(\alpha)) \in \delta((x, c_{\sigma_x}(\alpha)), a) \text{ such that } z \sim w \end{cases}$$

Let us consider the case (1). As $(z, c_{\sigma_z}(\alpha)) \in \delta((x, c_{\sigma_x}(\alpha)), a)$, by definition of δ we know that $\overline{z} = a$ and $d_z(c_{\sigma_x}(\alpha)) = c_{\sigma_z}(\alpha)$. By Proposition 9 we can conclude that $\overline{c_{\sigma_z}(\alpha)} \in \partial_a(\overline{c_{\sigma_x}(\alpha)})$. By hypothesis, we know that $\overline{c_{\sigma_x}(\alpha)} \equiv \overline{c_{\sigma_y}(\alpha)}$, thus $\overline{c_{\sigma_z}(\alpha)} \in \partial_a(\overline{c_{\sigma_y}(\alpha)})$. But by Proposition 9 there exists a w such that the following is true:

- $\overline{w} = a$, which implies $d_w(c_{\sigma_y}(\alpha)) = c_{\sigma_w}(\alpha)$ and
- $\overline{d_w(c_{\sigma_y}(\alpha))} \equiv \overline{d_z(c_{\sigma_z}(\alpha))}$ which implies $\overline{c_{\sigma_w}(\alpha)} \equiv \overline{c_{\sigma_z}(\alpha)}$

The proof for (2) is similar. We also need to prove that $\forall a \in \Sigma (x, c_{\sigma_x}(\alpha)) \sim (y, c_{\sigma_y}(\alpha)) \implies \lambda((x, c_{\sigma_x}(\alpha))) = \lambda((y, c_{\sigma_y}(\alpha)))$, which is obvious.

□

3.2 Finite Languages

In this section, we consider normalized regular expressions without the Kleene star operator, i.e. that represent finite languages. The set of these regular expressions α over Σ can be defined by the grammar below:

$$\begin{array}{rcl} \alpha & := & \emptyset \mid \varepsilon \mid \beta \mid \gamma \mid \sigma \in \Sigma \\ \beta & := & \beta_1 + \varepsilon \mid \beta_0 + \gamma \mid \beta_0 + \sigma \\ \beta_0 & := & \gamma \mid \sigma \in \Sigma \mid \beta_0 + \beta_0 \\ \gamma & := & \gamma_0\gamma_0 \mid \gamma_0\gamma \end{array} \qquad \begin{array}{rcl} \gamma_0 & := & (\beta) \mid \sigma \in \Sigma \\ \beta_1 & := & \gamma_2 \mid \sigma \in \Sigma \mid \beta_1 + \beta_1 \\ \gamma_2 & := & \gamma_0\beta_1 \mid \beta_1\gamma_0 \end{array}$$

To know that this grammar is correct, we must prove that $L(\alpha) = L_r$, where L_r is the set of normalized regular expressions. To prove this equivalence, we must consider the relation in both directions. First, the grammar must produce only regular repressions found in L_r. Second, every regular expression in L_r must be produced by the grammar.

Let us prove that $L(\alpha) \subset L_r$. The first grammar rule α produce the basic regular expressions $\emptyset$, ϵ and σ, which are obviously normalized regular expressions. This rule also produce disjunctions (rule β) and conjunctions (rule γ). The conjunctions are defined in the usual way, thus the rule γ is obvious. However the disjunctions, can only be defined in the usual way if none of the terms is ϵ. If one of the terms of the disjunction is ϵ) then ϵ can not belong to the language of the other disjunction term. Therefore, the rule β_1 produces regular expressions for which ϵ does not belong to its language. These regular expressions can be σ, disjunctions of other regular expressions without ϵ in its language ($\beta_1 + \beta_1$), or conjunctions in which at least one term has not ϵ in its language (rule γ_2). It is not difficult to see that $\gamma_2 \to \gamma_0\beta_1 \to \gamma_0\gamma_0\beta_1 \to \gamma_0\gamma_0\beta_1\gamma_0 \to \ldots$, i.e. $\gamma_2 \Rightarrow^\star \gamma_0^\star\beta_1\gamma_0^+$ or $\gamma_2 \Rightarrow^\star \gamma_0^+\beta_1\gamma_0^\star$. As γ_0 represents a σ or a disjunction with or without ϵ is not difficult to conclude that γ_2 represents a conjunction with at least one term which has not ϵ in its language. Therefore, every regular expression generated by the grammar is in L_r.

Considering L_1 as the set of regular expressions for which ϵ does not belong to its language, let us prove that $L_1 \subset L(\beta_1)$. We will proceed by induction on the structure of $r \in L_r$. If $r = \sigma$ it is obvious that β_1 produces r. Let us suppose that for any subexpressions r_i of r, if $\epsilon \in r_i$ then $\beta_1 \Rightarrow^\star r_i$, and if $r_i \in L_r \setminus \{\emptyset, \epsilon\}$ then $\gamma_0^+ \Rightarrow^\star r_i$. Thus, if $r = r_1 + r_2$, we know that $\epsilon \notin L(r_1)$ and $\epsilon \notin r_2$ because $r \in L_1$. So $\beta_1 \to \beta_1 + \beta_1 \Rightarrow^\star r_1 + r_2$. If $r = r_1r_2$, we know that either $\epsilon \notin L(r_1)$ or $\epsilon \notin r_2$. Therefore, if $\epsilon \notin L(r_1)$ then $\beta_1 \to \beta_1\gamma_0^+ \Rightarrow^\star r_1r_2$; and if $\epsilon \notin L(r_2)$ then $\beta_1 \to \gamma_0^+\beta_1 \Rightarrow^\star r_1r_2$. So any regular expression in L_1 is also in $L(\beta_1)$. From this, it is obvious that $L_r \subset L(\alpha)$.

The regular expressions represented by this grammar are named *finite regular expressions.* The following results characterize NFAs that are $\mathcal{A}_{pd}$ automaton.

Proposition 11. *The $\mathcal{A}_{pd}(\alpha) = (\mathsf{PD}(\alpha), \Sigma, \delta_\alpha, \alpha, F_\alpha)$ automaton of any finite regular expression $\alpha \not\equiv \emptyset$ has the following properties:*

1. *The state ε always exists and it is a final state;*
2. *The state ε is reachable from any other state;*
3. *All other final states, $q \in F_\alpha \setminus \{\varepsilon\}$, are of the form $(\alpha_1 + \varepsilon)\ldots(\alpha_n + \varepsilon)$;*
4. *$|F_\alpha| \leq |\alpha|_\varepsilon + 1$;*
5. *The size of each element of $\mathsf{PD}(\alpha)$ is not greater than $|\alpha|$.*

Proof. We use the inductive construction of $\mathcal{A}_{pd}(\alpha)$.

1. For the base cases this is obviously true. If α is $\gamma + \beta$, then $\pi(\alpha) = \pi(\gamma) \cup \pi(\beta)$. As $\varepsilon \in \pi(\gamma)$ and $\varepsilon \in \pi(\beta)$, by inductive hypothesis, then $\varepsilon \in \pi(\alpha)$. If α is $\gamma\beta$, then $\pi(\alpha) = \pi(\gamma)\beta \cup \pi(\beta)$. As $\varepsilon \in \pi(\beta)$, $\varepsilon \in \pi(\alpha)$.

2. If α is ε or σ it is obviously true. Let α be $\gamma + \beta$. The states of $\mathcal{A}_{pd}(\alpha)$ are $\{\alpha\} \cup \pi(\gamma) \cup \pi(\beta)$. By construction, there exists at least a transition from the state α to a (distinct) state in $\pi(\gamma) \cup \pi(\beta)$. Let α be $\gamma\beta$. The states of $\mathcal{A}_{pd}(\alpha)$ are $\{\alpha\} \cup \pi(\gamma)\beta \cup \pi(\beta)$. For $\beta' \in \{\beta\} \cup \pi(\beta)$, $\exists w_\beta\ \varepsilon \in \partial_{w_\beta}(\beta')$. In the same way, for $\gamma' \in \{\gamma\} \cup \pi(\gamma)$, $\exists w_\gamma\ \varepsilon \in \partial_{w_\gamma}(\gamma')$. Thus, for $\alpha' = \gamma'\beta \in \pi(\gamma)\beta$, we can conclude that $\varepsilon \in \partial_{w_\gamma w_\beta}(\alpha')$. From the state α we can reach the state ε because the transitions leaving it go to states which reach the state ε.

3. It is obvious, because final states must accept ε.

4. For the base cases it is obviously true. Let α be $\gamma+\beta$. We know that $|\alpha|_\varepsilon = |\gamma|_\varepsilon + |\beta|_\varepsilon$, $|F_\alpha| \leq |F_\gamma| + |F_\beta| - 1$, and that $\varepsilon(\alpha) = \varepsilon$ if either $\varepsilon(\gamma)$ or $\varepsilon(\beta)$ are ε. Then $|F_\alpha| \leq |\gamma|_\varepsilon + |\beta|_\varepsilon + 1 \leq |\alpha|_\varepsilon + 1$. If α is $\gamma\beta$ we know also that $|\alpha|_\varepsilon = |\gamma|_\varepsilon + |\beta|_\varepsilon$ and that $\varepsilon(\alpha) = \varepsilon$ if $\varepsilon(\gamma)$ and $\varepsilon(\beta)$ are ε. If $\varepsilon(\beta) = \varepsilon$, then $|F_\alpha| \leq |F_\gamma| + |F_\beta| - 1$. Otherwise, $|F_\alpha| = |F_\beta|$. We have, in the both cases, $|F_\alpha| \leq |\gamma|_\varepsilon + |\beta|_\varepsilon + 1 \leq |\alpha|_\varepsilon + 1$.

5. If α is ε or σ it is obvious that the proposition is true. Let α be $\gamma + \beta$. For all $\alpha_i \in \pi(\alpha) = \pi(\gamma) \cup \pi(\beta)$, $|\alpha_i| \leq |\gamma|$ or $|\alpha_i| \leq |\beta|$, and thus $|\alpha_i| \leq |\alpha|$. If α is $\gamma\beta$, then $\pi(\alpha) = \pi(\gamma)\beta \cup \pi(\beta)$. For $\gamma_i \in \pi(\gamma)$, $|\gamma_i| \leq \gamma$. If $\alpha_i \in \pi(\gamma)\beta$, $\alpha_i = \gamma_i\beta$ and $|\alpha_i| \leq |\gamma| + |\beta| \leq |\alpha|$. If $\alpha_i \in \pi(\beta)$, $|\alpha_i| \leq |\beta| \leq |\alpha|$.

□

Caron and Ziadi [7] characterized the position automaton in terms of the properties of the underlying digraph. We consider a similar approach to characterize the $\mathcal{A}_{pd}$ for finite languages. We restrict the analysis to acyclic NFAs. We first observe that $\mathcal{A}_{pos}$ are series-parallel automata [19] which is not the case for all $\mathcal{A}_{pd}$ as can be seen considering $\mathcal{A}_{pd}(a(ac+b)+bc)$.

Let $A = (Q, \Sigma, \delta, q_0, F)$ be an acyclic NFA. A is an *hammock* if it has the following properties. If $|Q| = 1$, A has no transitions. Otherwise, there exists an unique $f \in F$ such that for any state $q \in Q$ one can find a path from q_0 to f going through q. The state q_0 is called the *root* and f the *anti-root*. The *rank* of a state $q \in Q$, named $rk(q)$, is the length of the longest word $w \in \Sigma^\star$ such that $\delta(q, w) \in F$. In an hammock, the anti-root has rank 0. Each state q of rank $r \geq 1$, has only transitions for states in smaller ranks and at least one transition for a state in rank $r - 1$.

Proposition 12. *For every finite regular expression α, $\mathcal{A}_{pd}(\alpha)$ is an hammock.*

Proof. If the partial derivative automaton has a unique state then it is the $A_{pd}(\varepsilon)$ or $A_{pd}(\emptyset)$ which has no transitions. Otherwise, for all $q \in \mathsf{PD}(\alpha)$ there exists at least one path from $q_0 = \alpha$ to q because $\mathcal{A}_{pd}(\alpha)$ is initially connected; also there exists at least one path from q to ε, the anti-root, by Proposition 11, item 2. □

Proposition 13. *An acyclic NFA $A = (Q, \Sigma, \delta, q_0, F)$ is a partial derivative automaton of some finite regular expression α, if the following conditions holds:*

1. *A is an hammock;*

2. *$\forall q, q' \in Q \; rk(q) = rk(q') \implies \exists \sigma \in \Sigma \; \delta(q, \sigma) \neq \delta(q', \sigma)$.*

Proof. First we give an algorithm that allows to associate to each state of an hammock A a regular expression. Then, we show that if the second condition holds, A is the $\mathcal{A}_{pd}(\alpha)$ where α is the RE associated to the initial state.

We label each state q with a regular expression $RE(q)$, considering the states by increasing rank order. We define for the anti-root f, $RE(f) = \varepsilon$. Suppose that all states of ranks less then n are already labelled. Let $q \in Q$ with $rk(q) = n$. For $\sigma \in \Sigma$, with $\delta(q, \sigma) = \{q_1, \ldots, q_m\}$ and $RE(q_i) = \beta_i$ we construct the regular expression $\sigma(\beta_1 + \cdots + \beta_m)$. Then,

$$RE(q) = \sum_{\sigma \in \Sigma} \sigma(\beta_1 + \cdots + \beta_m)$$

where we omit all $\sigma \in \Sigma$ such that $\delta(q, \sigma) = \emptyset$. We have, $RE(q_0) = \alpha$

To show that if A satisfies condition 2. then $A \simeq \mathcal{A}_{pd}(\alpha)$, we need to prove that $RE(q) \not\equiv RE(q')$ for all $q, q' \in Q$ with $q \neq q'$. We prove by induction on the rank. For rank 0, it is obvious. Suppose that all states with rank $m < n$ are labelled by different regular expressions. Let $q \in Q$, with $rk(q) = n$. We must prove that $RE(q) \not\equiv RE(q')$ for all q' with $rk(q') \leq n$. Suppose that $rk(q) = rk(q')$, $RE(q) = \sigma_1(\alpha_1 + \cdots + \alpha_n) + \cdots + \sigma_i(\beta_1 + \cdots + \beta_m)$, and $RE(q') = \sigma'_1(\alpha'_1 + \cdots + \alpha'_{n'}) + \cdots + \sigma'_j(\beta'_1 + \cdots + \beta'_{m'})$. We know that $\exists \sigma \delta(q, \sigma) \neq \delta(q', \sigma)$. Suppose that $\sigma = \sigma_1 = \sigma'_1$. Then we know that $\exists t, t'\ \alpha_t \not\equiv \alpha'_{t'}$, thus $RE(q) \not\equiv RE(q')$. If $rk(q) > rk(q')$, then there exists a $w \in \Sigma^\star$ with $|w| = n$ such that $\delta(q, w) \cap F \neq \emptyset$ and $\delta(q', w) \cap F = \emptyset$. Thus $RE(q) \not\equiv RE(q')$. $\square$

3.3 Comparing $\mathcal{A}_{pd}$ and $\mathcal{A}_{pos}/_{\equiv_b}$

As we already mentioned, there are many (normalized) regular expressions α for which $\mathcal{A}_{pd}(\alpha) \simeq \mathcal{A}_{pos}(\alpha)/_{\equiv_b}$. But, even for REs representing finite languages that is not always true. Taking, for example, $\tau_1 = a(a+b)c + b(ac+bc) + a(c+c)$, we have $\mathsf{PD}(\tau_1) = \{\tau_1, ac+bc, (a+b)c, c+c, c, \varepsilon\}$, $F_{pd} = \{\varepsilon\}$, $\delta_{pd}(\tau_1, a) = \{(a+b)c, c+c\}$, $\delta_{pd}(\tau_1, b) = \{ac+bc\}$, $\delta_{pd}(ac+bc, a) = \delta_{pd}(ac+bc, b) = \delta_{pd}((a+b)c, a) = \delta_{pd}((a+b)c, b) = \{c\}$ and $\delta_{pd}(c+c, c) = \delta_{pd}(c, c) = \{\varepsilon\}$. One can see that $c \equiv_b (c+c)$ and $(ac+bc) \equiv_b (a+b)c$. Thus, $\mathcal{A}_{pos}(\tau_1)/_{\equiv_b}$ has two states less than $\mathcal{A}_{pd}(\tau_1)$. The states that are bisimilar are equivalent modulo the $+$ idempotence and left-distributivity. It is also easy to see that two states are bisimilar if they are equivalent modulo $+$ associativity or $+$ commutativity.

Considering an order $<$ on Σ and that $\cdot < +$, we can extend $<$ to REs. Then, the following rewriting system is confluent and terminating:

$$
\begin{array}{rcrl}
\alpha + (\beta + \gamma) \to (\alpha + \beta) + \gamma & & & (+\ \textbf{A}\text{ssociativity}) \\
\alpha + \beta \to \beta + \alpha & & \text{if } \beta < \alpha & (+\ \textbf{C}\text{ommutativity}) \\
\alpha + \alpha \to \alpha & & & (+\ \textbf{I}\text{dempotence}) \\
(\alpha\beta)\gamma \to \alpha(\beta\gamma) & & & (.\ \textbf{A}\text{ssociativity}) \\
(\alpha + \gamma)\beta \to \alpha\beta + \gamma\beta & & & (\textbf{L}\text{eft distributivity}).
\end{array}
$$

A (normalized) regular expression α that can not be rewritten anymore by this system is called an *irreducible regular expression modulo* ACIAL.

Remark 1. *An irreducible regular expression modulo* ACIAL α *is of the form:*

$$w_1 + \ldots + w_n + w'_1\alpha_1 + \ldots + w'_m\alpha_m \tag{8}$$

where w_i, w'_j *are words for* $1 \leq i \leq n$, $1 \leq j \leq m$, *and* α_j *are expressions of the same form of* α, *for* $1 \leq j \leq m$. *For for each normalized RE without the Kleene star operator, there exits a unique normal form.*

For example, considering $a < b < c$, the normal form for the RE τ_1 given above is $\tau_2 = ac + a(ac+bc) + b(ac+bc)$ and $\mathcal{A}_{pd}(\tau_2) \simeq \mathcal{A}_{pos}(\tau_2)/_{\equiv_b}$. As we will see next, for normal forms this isomorphism always holds.

The following lemmas are needed to prove the main result.

Lemma 14. *For* $\sigma \in \Sigma$, *the function* ∂_σ *is closed modulo* ACIAL.

Proof. We know that α has the form $w_1+\ldots+w_n+w'_1\alpha_1+\ldots+w'_i\alpha_m$, where $w_i = \sigma v_i, v_i \in \Sigma^\star$, $w'_j = \sigma v'_j, v'_j \in \Sigma^\star$, $i \in \{1, \cdots, n\}$, $j \in \{1, \cdots, m\}$. Thus, $\forall \sigma \in \Sigma\ \partial_\sigma(\alpha) = \partial_\sigma(w_1) \cup \cdots \cup \partial_\sigma(w_n) \cup \partial_\sigma(w'_1)\alpha_1 \cup \cdots \cup \partial_\sigma(w'_i)\alpha_m$, where $\partial_\sigma(w_i) = v_i$ and $\partial_\sigma(w'_j)\alpha_j = v'_j\alpha_j$. Then it is obvious that the both possible results are irreducible modulo ACIAL. Thus the proposition holds. □

Lemma 15. *For $w, w' \in \Sigma^\star$,*

1. $(\forall \sigma \in \Sigma)\ |\partial_\sigma(w)| \leq 1$.
2. $w \neq w' \implies (\forall \sigma \in \Sigma)\ \partial_\sigma(w) \neq \partial_\sigma(w') \ \vee\ \partial_\sigma(w) = \partial_\sigma(w') = \emptyset$.
3. $(\forall \sigma \in \Sigma)\partial_\sigma(w\alpha) = \partial_\sigma(w)\alpha = \{w'\alpha\}$, *if* $w = \sigma w'$.

Proof.

1. Let $w = \sigma w'$. Then $\partial_\sigma(w) = \partial_\sigma(\sigma w') = \{w'\}$. For $\sigma \neq \sigma'$, $\partial_{\sigma'}(w) = \emptyset$.
2. We need to consider three cases:
   (a) if $\sigma \notin \mathsf{First}(w)$ and $\sigma \notin \mathsf{First}(w')$ then $\partial_\sigma(w) = \emptyset$ and $\partial_\sigma(w') = \emptyset$.
   (b) if $\sigma \in \mathsf{First}(w)$ and $\sigma \notin \mathsf{First}(w')$ then $\partial_\sigma(w) \neq \emptyset$ and $\partial_\sigma(w') = \emptyset$.
   (c) if $\sigma \in \mathsf{First}(w)$, $\sigma \in \mathsf{First}(w')$ and $w = \sigma v$, $w' = \sigma v'$ then $v \neq v'$. As $\partial_\sigma(w) = v$ and $\partial_\sigma(w') = v'$ then $\partial_\sigma(w) \neq \partial_\sigma(w')$.
3. Let $w = \sigma w'$. Then $\partial_\sigma(w\alpha) = \partial_\sigma(w)\alpha = \partial_\sigma(\sigma w')\alpha = \{w'\alpha\}$. For $\sigma \neq \sigma'$, $\partial_{\sigma'}(w\alpha) = \emptyset$.

□

Proposition 16. *Given α and β irreducible finite regular expressions modulo* ACIAL*,*

$$\alpha \not\equiv \beta \implies \exists \sigma \in \Sigma\ \partial_\sigma(\alpha) \neq \partial_\sigma(\beta).$$

Proof. Let $\alpha \not\equiv \beta$. We know that $\alpha = w_1 + \cdots + w_n + w'_1\alpha_1 + \cdots + w'_m\alpha_m$ and $\beta = x_1 + \cdots + x_{n'} + x'_1\beta_1 + \cdots + x'_{m'}\beta_{m'}$. As we know that $(\forall \sigma \in \Sigma)\ |\partial_\sigma(w)| \leq 1$, we denote $\partial_\sigma(w)$ by $(w)^{-1}_\sigma$. The sets of partial derivatives of α and β w.r.t a $\sigma \in \Sigma$ can be written as:

$$\begin{aligned}(\alpha)^{-1}_\sigma &= A \cup (w_{i_1})^{-1}_\sigma \cup \cdots \cup (w_{i_j})^{-1}_\sigma \cup (w'_{l_1})^{-1}_\sigma\alpha_{l_1} \cup \cdots \cup (w'_{l_t})^{-1}_\sigma\alpha_{l_t},\\ (\beta)^{-1}_\sigma &= A \cup (x_{i'_1})^{-1}_\sigma \cup \cdots \cup (x_{i'_{j'}})^{-1}_\sigma \cup (x'_{l'_1})^{-1}_\sigma\beta_{l'_1} \cup \cdots \cup (x'_{l'_{t'}})^{-1}_\sigma\beta_{l'_{t'}},\end{aligned}$$

where the set A contains all partial derivatives φ such that $\varphi \in (\gamma)^{-1}_\sigma$ if, and only if, γ is a common summand of α and β, i.e. if $\gamma \equiv w_i \equiv x_j$ or $\gamma \equiv w'_l\alpha_l \equiv x'_k\beta_k$ for some i, j, l, and k. Without loss of generality, consider the following three cases:

1. If $i_1 \neq 0$ and $i'_1 \neq 0$, we know that for $k \in \{i'_1, \ldots, i'_{j'}\}$, $w_{i_1} \neq x_k$ and, by Lemma 15, $(w_{i_1})^{-1}_\sigma \neq (x_k)^{-1}_\sigma$. And also, by Lemma 15, $(w_{i_1})^{-1}_\sigma \neq (x'_k)^{-1}_\sigma\beta_k$, for $k \in \{l'_1, \ldots, l'_{t'}\}$. Thus, $(w_i)^{-1}_\sigma \cap (\beta)^{-1}_\sigma = \emptyset$.
2. If $i_1 \neq 0$ and $i'_j = 0$, this case corresponds to the second part of the previous one.
3. If $i_j = i'_{j'} = 0$, for $k \in \{l'_1, \ldots, l'_{t'}\}$, we have $w'_{l_1}\alpha_{l_1} \neq x'_k\alpha_k$ and then either $w'_{l_1} \neq x'_k$ or $\alpha_{l_1} \neq \beta_k$. If $w'_{l_1} \neq x'_k$ then $(w'_{l_1})^{-1}_\sigma \neq (x'_k)^{-1}_\sigma$ and thus $(w'_{l_1})^{-1}_\sigma\alpha_{l_1} \neq (x'_k)^{-1}_\sigma\alpha_k$. If $\alpha_{l_1} \neq \beta_k$ it is obvious that $(w'_l)^{-1}_\sigma\alpha_l \neq (x'_k)^{-1}_\sigma\alpha_k$. Thus, $(w'_{l_1})^{-1}_\sigma\alpha_{l_1} \cap (\beta)^{-1}_\sigma = \emptyset$.

□

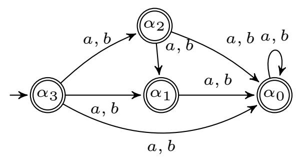
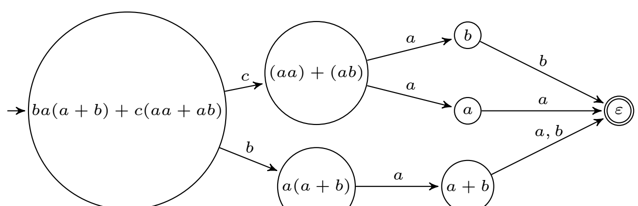


Figure 4: $\mathcal{A}_{pd}((a+b+\varepsilon)(a+b+\varepsilon)(a+b+\varepsilon)(a+b)^\star)$

Theorem 1. *Let α be a irreducible finite regular expression modulo* ACIAL*. Then, $\mathcal{A}_{pd}(\alpha) \simeq \mathcal{A}_{pos}(\alpha)/_{\equiv_b}$.*

Proof. Let $\mathcal{A}_{pd}(\alpha) = (\mathsf{PD}(\alpha), \Sigma, \delta_{pd}, \alpha, F_{pd})$. We want to prove that no pair of states of $\mathcal{A}_{pd}(\alpha)$ is bisimilar. As in Proposition 13, we proceed by induction on the rank of the states. The only state in rank 0 is ε, for which the proposition is obvious. Suppose that all pair of states with rank $m < n$ are not bisimilar. Let $\gamma, \beta \in \mathsf{PD}(\alpha)$ with $n = rk(\gamma) \geq rk(\beta)$. Then, there exists $\gamma' \in \partial_\sigma(\gamma)$ that is distinct of every $\beta' \in \partial_\sigma(\beta)$, by Proposition 16. Because $rk(\beta') < n$ and $rk(\gamma') < n$, by inductive hypothesis, $\gamma' \not\equiv_b \beta'$. Thus $\gamma \not\equiv_b \beta$. $\square$

Despite $\mathcal{A}_{pd}(\alpha) \simeq \mathcal{A}_{pos}(\alpha)/_{\equiv_b}$, for irreducible REs modulo ACIAL, these NFAs are not necessarily minimal. For example, if $\tau_3 = ba(a+b) + c(aa+ab)$, both NFAs have seven states, as can be seen in figure below, and a minimal equivalent NFA has four states.



Finally, note that for general regular expressions representing finite languages, $\mathcal{A}_{pos}(\alpha)/_{\equiv_b}$ can be arbitrarily more succinct than $\mathcal{A}_{pd}$. For example, considering the family of REs

$$\alpha_n = aa_1 + a(a_1+a_2) + a(a_1+a_2+a_3) + \ldots + a(a_1+a_2+\ldots+a_n)$$

the $\mathcal{A}_{pd}(\alpha_n)$ has $n+2$ states and $\mathcal{A}_{pos}(\alpha)/_{\equiv_b}$ has three states independently of n.

3.4 General Regular Languages

If we consider general regular expressions with the Kleene star operator, it is easy to find REs α such that $\mathcal{A}_{pd}(\alpha) \not\simeq \mathcal{A}_{pos}(\alpha)/_{\equiv_b}$. This is true even if $\mathcal{A}_{pos}(\alpha)$ is a DFA, i.e. if α is one-unambiguous [5]. For example, for $\alpha = aa^\star + b(\varepsilon + aa^\star)$ the $\mathcal{A}_{pd}(\alpha)$ has one more state than $\mathcal{A}_{pos}(\alpha)/_{\equiv_b}$. Ilie and Yu [15] presented a family of REs

$$\alpha_n = (a+b+\varepsilon)(a+b+\varepsilon)\ldots(a+b+\varepsilon)(a+b)^\star,$$

where $(a+b+\varepsilon)$ is repeated n times, for which $\mathcal{A}_{pd}(\alpha_n)$ has $n+1$ states and $\mathcal{A}_{pos}(\alpha_n)/_{\equiv_b}$ has one state independently of n. Considering $n = 3$ the $\mathcal{A}_{pd}(\alpha_3)$ are represented in Figure 4.

In concurrency theory, the characterization of regular expressions for which equivalent NFAs are bisimilar has been extensively studied. Baeten et. al [2] defined a normal form that corresponds to the normal form (8), in the finite case. For regular expressions with Kleene

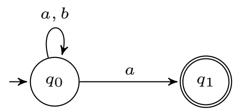


Figure 5: $\mathcal{A}_{pos}(\tau)/{\equiv_b}$.

star operator the normal form defined by those authors is neither irreducible nor unique. In that case, we can find regular expressions α in normal form such that $\mathcal{A}_{pd}(\alpha) \not\simeq \mathcal{A}_{pos}(\alpha)/{\equiv_b}$. For example, for $\tau = (ab^\star + b)^\star$ the $\mathcal{A}_{pd}(\tau)$ has three states, as seen before in Figure 2, and $\mathcal{A}_{pos}(\tau)/{\equiv_b}$ has two states, as shown in Figure 5. Other example is $\tau_4 = a(\varepsilon + aa^\star) + ba^\star$, where $|\mathsf{PD}(\tau_4)| = 3$, and in $\mathcal{A}_{pos}(\tau_4)/{\equiv_b}$ a state is saved because $(\varepsilon + aa^\star) \equiv_b a^\star$. This corresponds to an instance of one of the axioms of Kleene algebra (for the star operator).

As no confluent or even terminating rewrite system modulo these axioms is known, for general REs it will be difficult to obtain a characterization similar to the one of Theorem 1.

Acknowledgements

This work was partially funded by the European Regional Development Fund through the programme COMPETE and by the Portuguese Government through the FCT under project PEst-C/MAT/UI0144/2013 and project FCOMP-01-0124-FEDER-020486.

Eva Maia was also funded by FCT grant SFRH/BD/78392/2011.

The authors would like to thank the valuable remarks and corrections suggested by the anonymous referees.

References

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

[2] Baeten, J.C.M., Corradini, F., Grabmayer, C.A.: A characterization of regular expressions under bisimulation. J. ACM 54(2) (Apr 2007)

[3] Berry, G., Sethi, R.: From regular expressions to deterministic automata. Theor. Comput. Sci. 48(1), 117–126 (Dec 1986)

[4] Broda, S., Machiavelo, A., Moreira, N., Reis, R.: On the average size of Glushkov and partial derivative automata. International Journal of Foundations of Computer Science 23(5), 969–984 (2012)

[5] Brüggemann-Klein, A.: Regular expressions into finite automata. Theoret. Comput. Sci. 48, 197–213 (1993)

[6] Brzozowski, J.A.: Derivatives of regular expressions. J. ACM 11(4), 481–494 (Oct 1964)

[7] Caron, P., Ziadi, D.: Characterization of Glushkov automata. Theoret. Comput. Sci. 233(1-2), 75–90 (2000)

[8] Champarnaud, J.M., Ouardi, F., Ziadi, D.: Follow automaton versus equation automaton. In: Ilie, L., Wotschke, D. (eds.) DCFS. vol. Report No. 619, pp. 145–153. Department of Computer Science, The University of Western Ontario, Canada (2004)

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

[10] Champarnaud, J.M., Ziadi, D.: Canonical derivatives, partial derivatives and finite automaton constructions. Theor. Comput. Sci. 289(1), 137–163 (2002)

[11] García, P., López, D., Ruiz, J., Alvarez, G.I.: From regular expressions to smaller nfas. Theor. Comput. Sci. 412(41), 5802–5807 (2011)

[12] Glushkov, V.M.: The abstract theory of automata. Russian Mathematical Surveys 16(5), 1–53 (1961)

[13] Gouveia, H., Moreira, N., Reis, R.: Small nfas from regular expressions: Some experimental results. CoRR abs/1009.3599 (2010)

[14] Ilie, L., Yu, S.: Follow automata. Inf. Comput. 186(1), 140–162 (2003)

[15] Ilie, L., Yu, S.: Reducing nfas by invariant equivalences. Theor. Comput. Sci. 306(1-3), 373–390 (Sep 2003)

[16] Kozen, D.C.: Automata and Computability. Springer (1997)

[17] McNaughton, R., Yamada, H.: Regular expressions and state graphs for automata. IEEE Transactions on Electronic Computers 9, 39–47 (1960)

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

[19] Moreira, N., Reis, R.: Series-parallel automata and short regular expressions. Fundam. Inform. 91(3-4), 611–629 (2009)

[20] Paige, R., Tarjan, R.E.: Three partition refinement algorithms. SIAM J. Comput. 16(6), 973–989 (1987)