On the Mother of All Automata: the Position Automaton

Sabine Broda[1], Markus Holzer[2], Eva Maia[1],
Nelma Moreira[1], and Rogério Reis[1]

[1] CMUP and DCC, Faculdade de Ciências da Universidado do Porto ⋆,
Rua do Campo Alegre, 1021, 4169-007 Porto, Portugal
{sbb,emaia,nam,rvr}@dcc.fc.up.pt
[2] Institut für Informatik, Universität Giessen,
Arndtstr. 2, 35392 Giessen, Germany
holzer@informatik.uni-giessen.de

Abstract. We contribute new relations to the taxonomy of different conversions from regular expressions to equivalent finite automata. In particular, we are interested in ordinary transformations that construct automata such as, the follow automaton, the partial derivative automaton, the prefix automaton, the automata based on pointed expressions recently introduced and studied, and last but not least the position, or Glushkov automaton ($\mathcal{A}_{\mathrm{POS}}$), and their double reversed construction counterparts. We deepen the understanding of these constructions and show that with the artefacts used to construct the Glushkov automaton one is able to capture most of them. As a byproduct we define a *dual* version $\mathcal{A}_{\overleftarrow{\mathrm{POS}}}$ of the position automaton which plays a similar role as $\mathcal{A}_{\mathrm{POS}}$ but now for the reverse expression. It turns out that although the conversion of regular expressions and reversal of regular expressions to finite automata seems quite similar, there are significant differences.

1 Introduction

It is well known that regular expressions define exactly the same languages as deterministic or nondeterministic finite automata. The conversion between these representations has been intensively studied for more than half a century—see, e.g., Gruber and Holzer [11] for a recent survey on this subject w.r.t. descriptional complexity. There are a few classical algorithms and variants thereof for converting finite automata into equivalent regular expressions and as shown in [17] all these approaches are more or less reformulations of the same underlying algorithmic idea, and they yield (almost) the same regular expressions. For the converse transformation, that is, the conversion of regular expressions into equivalent finite automata, the situation is much more diverse, since the algorithmic underlying ideas already are different. Nevertheless, for some of the

⋆ Authors partially supported by CMUP (UID/MAT/00144/2013), which is funded by FCT (Portugal) with national (MEC) and European structural funds through the programs FEDER, under the partnership agreement PT2020.

algorithms the constructed automata can still be related to each other by determinisation and/or quotients w.r.t. equivalence relations. For instance, by Ilie and Yu [12] it was shown that for a regular expression α the follow automaton $\mathcal{A}_{\mathrm{F}}(\alpha)$ is isomorphic ($\simeq$) to the quotient of the position or Glushkov [10] automaton $\mathcal{A}_{\mathrm{POS}}(\alpha)$ w.r.t. the relation $\equiv_{\mathrm{F}}$, that is, $\mathcal{A}_{\mathrm{F}}(\alpha) \simeq \mathcal{A}_{\mathrm{POS}}(\alpha)/\equiv_{\mathrm{F}}$. Another relation is that the determinisation of the position automaton $\mathcal{A}_{\mathrm{POS}}(\alpha)$ is the McNaughton and Yamada [14] automaton $\mathcal{A}_{\mathrm{MY}}(\alpha)$, or in mathematical notation $\mathrm{D}(\mathcal{A}_{\mathrm{POS}}(\alpha)) = \mathcal{A}_{\mathrm{MY}}(\alpha)$. From the variety of contructions from regular expressions to equivalent finite automata these are only two examples where the position automaton plays a central role.

We contribute further relations to the taxonomy of conversions from regular expressions to finite automata—see Figure 1 on page 11. Arrows, that are displayed in bold in that figure correspond to new contributions in this paper. Provenance of results that are not original is well indicated. Besides the above mentioned follow automaton $\mathcal{A}_{\mathrm{F}}$ we also consider the partial derivative automaton $\mathcal{A}_{\mathrm{PD}}$ of Mirkin [15] and Antimirov [2], the prefix automaton $\mathcal{A}_{\mathrm{Pre}}$ of Yamamoto [19], and contructions based on a recent approach of Asperti et al. [3] and by Nipkow and Traytel [16] by pointed expressions that lead to the mark after and mark before automata $\mathcal{A}_{\mathrm{MA}}$ and $\mathcal{A}_{\mathrm{MB}}$, respectively. Pointed expressions are an alternative representation of sets of positions. For the follow automaton $\mathcal{A}_{\mathrm{F}}$ we show that it can be directly computed from the expression by labelling states not with positions but with their Follow sets and their finality, and that the quotient of the determinised follow automaton w.r.t. a right-invariant relation $\equiv_s$, which is a generalization of the $\equiv_{\mathrm{F}}$-relation, leads to the mark before automaton $\mathcal{A}_{\mathrm{MB}}$. It is known that $\mathcal{A}_{\mathrm{MA}}$ is isomorphic to the Yamada-McNaughton automaton $\mathcal{A}_{\mathrm{MY}}$, which is proven to be the determinisation of the prefix automaton $\mathcal{A}_{\mathrm{Pre}}$. From $\mathcal{A}_{\mathrm{MA}}$ to $\mathcal{A}_{\mathrm{MB}}$ we present a homomorphism, showing that $\mathcal{A}_{\mathrm{MA}}$ cannot be smaller than $\mathcal{A}_{\mathrm{MB}}$—compare with [16].

When considering pointed expressions with only one point marking we obtain the position automaton in case of the mark after interpretation, while the other interpretation leads us to a *dual* version $\mathcal{A}_{\overleftarrow{\mathrm{POS}}}$ of the position automaton. We show that the double reverse construction $\mathcal{A}_{\mathrm{POS}}(\alpha^{\mathrm{R}})^{\mathrm{R}}$ is isomorphic to $\mathcal{A}_{\overleftarrow{\mathrm{POS}}}(\alpha)$ and that the determinisation of $\mathcal{A}_{\overleftarrow{\mathrm{POS}}}(\alpha)$ yields $\mathcal{A}_{\mathrm{MB}}(\alpha)$. Our study provides evidence that $\mathcal{A}_{\overleftarrow{\mathrm{POS}}}$ plays a similar role as $\mathcal{A}_{\mathrm{POS}}$, but for the reverse expression α^{R} instead of the expression α. This is supported by the fact that $\mathrm{D}(\mathcal{A}_{\mathrm{POS}}(\alpha^{\mathrm{R}})^{\mathrm{R}})$ is isomorphic to $\mathrm{D}(\mathcal{A}_{\mathrm{F}}(\alpha^{\mathrm{R}})^{\mathrm{R}})$ and $\mathrm{D}(\mathcal{A}_{\mathrm{PD}}(\alpha^{\mathrm{R}})^{\mathrm{R}})$. It is worth mentioning that the double reverse automata $\mathcal{A}_{\mathrm{Pre}}(\alpha^{\mathrm{R}})^{\mathrm{R}}$ and its determinisation $\mathrm{D}(\mathcal{A}_{\mathrm{Pre}}(\alpha^{\mathrm{R}})^{\mathrm{R}})$ get out of the line since the latter automaton turns out to be *not* isomorphic to $\mathrm{D}(\mathcal{A}_{\mathrm{POS}}(\alpha^{\mathrm{R}})^{\mathrm{R}})$. This shows, that although the taxonomy of "ordinary" conversions and double reversal conversions is quite similar, there are subtle differences that break the symmetry. Most proofs of our results are based on Glushkov's position concept which turns out to be highly valuable and can be used to describe automata constructions that look different at first sight, not only for the implementation use but also from the theoretical perspective. Due to space limitations, most proofs are omitted.

2 Preliminaries

In this section we review some basic definitions about regular expressions and finite automata and fix notation. Given an alphabet (finite set of *letters* or *alphabet symbols*) Σ, the set RE of *regular expressions*, α, over Σ is defined inductively by: $\emptyset$, ε and every letter σ_i is a regular expression, when α and α' are regular expressions then $(\alpha + \alpha')$, $(\alpha \cdot \alpha')$, and $(\alpha^\star)$ are regular expressions. The *language* associated to α is denoted by $\mathcal{L}(\alpha)$ and defined as usual. The *alphabetic size* $|\alpha|_\Sigma$ is its number of letters. We denote the subset of Σ containing the symbols that occur in α by Σ_α. We define $\varepsilon(\alpha)$ by $\varepsilon(\alpha) = \varepsilon$ if $\varepsilon \in \mathcal{L}(\alpha)$, and $\varepsilon(\alpha) = \emptyset$ otherwise. By abuse of notation, we consider $\varepsilon S = S$ and $\emptyset S = \emptyset$, for any set S. A *nondeterministic finite automaton* (NFA) is a five-tuple $A = \langle Q, \Sigma, \delta, I, F\rangle$ where Q is a finite set of states, Σ is a finite alphabet, $I \subseteq Q$ is the set of initial states, $F \subseteq Q$ is the set of final states, and $\delta : Q \times (\Sigma \cup \{\varepsilon\}) \to 2^Q$ is the transition function. We consider the size of an NFA as its number of states. An NFA that has transitions labelled with ε is an ε-NFA. In this paper, excepted when explicitly mentioned, we will consider NFAs without ε transitions. The transition function can be extended to words and to sets of states in the natural way. When $I = \{q_0\}$, we use $I = q_0$. We define the *finality* function ε on Q by $\varepsilon(q) = \varepsilon$ if $q \in F$ and $\varepsilon(q) = \emptyset$, otherwise. For $S \subseteq Q$ we have $\varepsilon(S) = \varepsilon$ iff there is some state $q \in S$ with $\varepsilon(q) = \varepsilon$, and $\varepsilon(S) = \emptyset$ otherwise. An NFA accepting a non-empty language is *trim* if every state is accessible from an initial state and every state leads to a final state. The language accepted by A is $\mathcal{L}(A) = \{\, w \in \Sigma^\star \mid \delta(I, w) \cap F \neq \emptyset \,\}$. Two automata are *equivalent* if they accept the same language. If two automata A and B are isomorphic, we write $A \simeq B$.

An NFA is *deterministic* (DFA) if $|\delta(q, \sigma)| \leq 1$, for all $(q, \sigma) \in Q \times \Sigma$, and $|I| = 1$. In this case, we simply write $\delta(p, a) = q$ instead of $\delta(p, a) = \{q\}$. We can convert an NFA A into an equivalent DFA $\mathrm{D}(A)$ by the *determinisation* operation D, using the well-known subset construction, where only subsets reachable from the initial subset of $\mathrm{D}(A)$ are used. Formally, $\mathrm{D}(A) = \langle Q_\mathrm{D}, \Sigma, \delta_\mathrm{D}, I_\mathrm{D}, F_\mathrm{D}\rangle$, where $Q_\mathrm{D} \subseteq 2^Q$, $I_\mathrm{D} = I$, $\delta_\mathrm{D}(S, \sigma) = \bigcup_{q\in S} \delta(q, \sigma)$ for $S \subseteq Q$, $\sigma \in \Sigma$, and $F_\mathrm{D} = \{\, S \in Q_\mathrm{D} \mid S \cap F \neq \emptyset \,\}$. Note that $S \in F_\mathrm{D}$ if and only if $\varepsilon(S) = \varepsilon$.

An equivalence relation $\equiv$ on Q is *right invariant* w.r.t. an NFA A if and only if: $\equiv \subseteq (Q - F)^2 \cup F^2$; and $\forall p, q \in Q, \sigma \in \Sigma$, if $p \equiv q$, then $\forall p' \in \delta(p, \sigma)\ \exists q' \in \delta(q, \sigma)$ such that $p' \equiv q'$. Given a set of states $S \subseteq Q$, we denote $S/\equiv\ = \{\, [q] \mid q \in S \,\}$. Note that $p \equiv q$ implies $\delta(p, \sigma)/\equiv\ = \delta(q, \sigma)/\equiv$, for $p, q \in Q$ and $\sigma \in \Sigma$. Furthermore, if A is deterministic, then $p \equiv q$ implies $\delta(p, \sigma) \equiv \delta(q, \sigma)$. If $\equiv$ is a right-invariant relation on Q, the *quotient automaton* $A/\equiv$ is given by $A/\equiv\ = \langle Q/\equiv, \Sigma, \delta/\equiv, [q_0], F/\equiv\rangle$, where $\delta/\equiv([p], \sigma) = \{\, [q] \mid q \in \delta(p, \sigma) \,\} = \delta(p, \sigma)/\equiv$. It is easy to see that $\mathcal{L}(A/\equiv) = \mathcal{L}(A)$. Given a right-invariant relation $\equiv$ w.r.t. an NFA A, we can consider the natural extension of $\equiv$ w.r.t. $\mathrm{D}(A)$, where for $X, Y \subseteq 2^Q$ we have $X \equiv Y$ if and only if $X/\equiv\ = Y/\equiv$. The following lemma relates determinisation with these right-invariant relations.

Lemma 1. $\mathrm{D}(A/\equiv) = \mathrm{D}(A)/\equiv$, *if* $\equiv$ *is a right-invariant relation w.r.t.* A.

3 The Position Automaton and "*the Rest*"

In this section we recall the definition of the position automaton and several related automata constructions. In particular, the determinisation of the position automaton, some ε-NFAs, derivative based constructions and the follow automaton are considered. We show that the latter can be obtained directly from the regular expression.

To decide if a word is represented by a regular expression, one can scan the symbols of the regular expression in a specific way. For instance, given $\alpha = a(bb + aba)^\star b$ the word *abbabab* can be obtained by scanning the first a, the two consecutive bs and then the second a, the third b, the third a, and the last b. This illustrates that uniquely identifying each letter of a regular expression is important for word recognition. Formally, given $\alpha \in \mathrm{RE}$, one can mark each occurrence of a letter σ with its position in α, considering reading it from left to right. The resulting regular expression is a *marked* regular expression $\overline{\alpha}$ with all symbols distinct and over the alphabet $\Sigma_{\overline{\alpha}}$. Then, a position $i \in [1, |\alpha|_\Sigma]$ corresponds to the symbol σ_i in $\overline{\alpha}$, and consequently to exactly one occurrence of σ in α. For instance, $\overline{\alpha} = a_1(b_2b_3 + a_4b_5a_6)^\star b_7$. The same notation is used for unmarking, $\overline{\overline{\alpha}} = \alpha$. Let $\mathsf{Pos}(\alpha) = \{1, 2, \ldots, |\alpha|_\Sigma\}$, and $\mathsf{Pos}_0(\alpha) = \mathsf{Pos}(\alpha) \cup \{0\}$.

Positions were used by Glushkov [10] to define an NFA equivalent to α, usually called the *position automaton* or Glushkov automaton ($\mathcal{A}_{\mathrm{POS}}(\alpha)$). Each state of the automaton, except for the initial state, corresponds to a position and there exists a transition from a position i to position j by a letter σ such that $\overline{\sigma_j} = \sigma$, if σ_i can be followed by σ_j in some word represented by $\overline{\alpha}$. More formally this reads as follows: the sets characterising the positions that can begin, end or be followed in words of $\mathcal{L}(\overline{\alpha})$ are, $\mathsf{First}(\alpha) = \{\, i \mid \sigma_i w \in \mathcal{L}(\overline{\alpha}) \,\}$, $\mathsf{Last}(\alpha) = \{\, i \mid w\sigma_i \in \mathcal{L}(\overline{\alpha}) \,\}$, and $\mathsf{Follow}(\alpha) = \{\, (i,j) \mid u\, \sigma_i\sigma_j v \in \mathcal{L}(\overline{\alpha}) \,\}$ respectively. We also define $\mathsf{Last}_0(\alpha) = \mathsf{Last}(\alpha) \cup \varepsilon(\alpha)\{0\}$. Furthermore, given $i \in \mathsf{Pos}(\alpha)$ and $S \in 2^{\mathsf{Pos}_0(\alpha)}$, let $\mathsf{Follow}(\alpha, i) = \{\, j \mid (i,j) \in \mathsf{Follow}(\alpha) \,\}$, $\mathsf{Follow}(\alpha, 0) = \mathsf{First}(\alpha)$ and $\mathsf{Follow}(\alpha, S) = \bigcup_{i \in S} \mathsf{Follow}(\alpha, i)$. The *position automaton* for α is

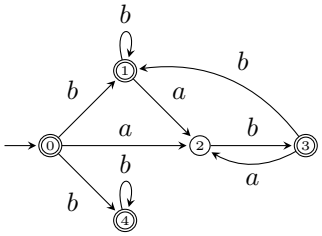
$$\mathcal{A}_{\mathrm{POS}}(\alpha) = \langle \mathsf{Pos}_0(\alpha), \Sigma, \delta_{\mathrm{POS}}, 0, \mathsf{Last}_0(\alpha) \rangle,$$

where $\delta_{\mathrm{POS}}(i, \sigma) = \{\, j \mid j \in \mathsf{Follow}(\alpha, i) \text{ and } \sigma = \overline{\sigma_j} \,\}$.

Proposition 2 ([10]). $\mathcal{L}(\mathcal{A}_{\mathrm{POS}}(\alpha)) = \mathcal{L}(\alpha)$.

The following example gives some intuition on the construction of the position automaton and its behaviour. Note that $\varepsilon(0) = \varepsilon(\alpha)$, and we will use either one or the other as it will be more convenient.

Example 3. Consider $\alpha = (b + ab)^\star + b^\star$ with $\overline{\alpha} = (b_1 + a_2b_3)^\star + b_4^\star$. Then, $\mathsf{First}(\alpha) = \{1, 2, 4\}$, $\mathsf{Last}_0(\alpha) = \{0, 1, 3, 4\}$ and $\mathsf{Follow}(\alpha) = \{(1,1), (1,2), (2,3), (3,1), (3,2), (4,4)\}$. The position automaton $\mathcal{A}_{\mathrm{POS}}$ for α is depicted below.



□

Note that each state, different from 0, in the position automaton corresponds to a symbol σ_i in $\overline{\alpha}$, where σ is the symbol just read. Thus, one can define a function Select that selects from a set of positions $S \subseteq \mathsf{Pos}(\alpha)$, those that correspond to a given letter, i.e., $\mathsf{Select} : 2^{\mathsf{Pos}(\alpha)} \times \Sigma \to 2^{\mathsf{Pos}(\alpha)}$ defined is by

$$\mathsf{Select}(S, \sigma) = \{\, i \mid i \in S \text{ and } \overline{\sigma_i} = \sigma \,\}.$$

Then, δ_{POS} can be defined by composing Follow with Select, i.e.,

$$\delta_{\mathrm{POS}}(i, \sigma) = \mathsf{Select}(\mathsf{Follow}(\alpha, i), \sigma). \tag{1}$$

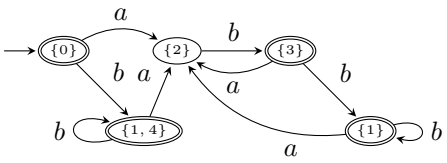
The same notion[3] was used by McNaughton and Yamada [14] to define an automaton which corresponds to the determinisation of the position automaton. With the definition of δ_{POS} in (1) and considering the determinisation algorithm, the McNaughton and Yamada DFA can be defined as

$$\mathcal{A}_{\mathrm{MY}}(\alpha) = \mathrm{D}(\mathcal{A}_{\mathrm{POS}}(\alpha)) = \langle Q_{\mathrm{MY}}, \Sigma, \delta_{\mathrm{MY}}, \{0\}, F_{\mathrm{MY}} \rangle,$$

where $Q_{\mathrm{MY}} \subseteq 2^{\mathsf{Pos}_0(\alpha)}$, $F_{\mathrm{MY}} = \{\, S \in Q_{\mathrm{MY}} \mid \varepsilon(S) = \varepsilon \,\}$ and for $S \in 2^{\mathsf{Pos}(\alpha)}$ and $\sigma \in \Sigma$, $\delta_{\mathrm{MY}}(S, \sigma) = \mathsf{Select}(\mathsf{Follow}(\alpha, S), \sigma)$.

Proposition 4 ([14]). $\mathcal{L}(\mathcal{A}_{\mathrm{MY}}(\alpha)) = \mathcal{L}(\alpha)$.

Example 5. Applying the McNaughton-Yamada construction to α from Example 3, we obtain the following DFA, $\mathcal{A}_{\mathrm{MY}}(\alpha)$:



□

In the forthcoming we review the Thompson like construction of the follow automaton $\mathcal{A}_{\mathrm{F}}$, which was introduced by Ilie and Yu in [12]. We show that one can directly construct this automaton by an appropriate state labeling inspired by the position automaton $\mathcal{A}_{\mathrm{POS}}$. Then we recall some constructions of automata from regular expressions based on derivatives and variants, such as, e.g., Brzozowski's construction by derivatives and the construction of Mirkin [15] and Antimirov [2] by partial derivatives. Here we focus on known results characterizing these automata as quotients of the position automaton.

[3] Some authors use slightly different notions of marking [8, 14], which have in common that each symbol in the marked expression corresponds to exactly one occurrence of a symbol in the original expression.

3.1 The Follow Automaton $\mathcal{A}_{\mathrm{F}}$

The most used conversion from regular expressions to equivalent ε-NFAs is the Thompson conversion [18], $\mathcal{A}_{\varepsilon\text{-T}}$. An improved use of ε-transitions lead to the definition of the ε-follow automaton [12]. From a Thompson automaton, if ε-transitions are eliminated in an adequate manner, the position automaton is obtained [1, 9]. Eliminating ε-transitions from the ε-follow automaton, the resulting automaton is the *follow automaton* $\mathcal{A}_{\mathrm{F}}(\alpha)$ which was introduced by Ilie and Yu [12] in 2003.

Proposition 6 ([12]). $\mathcal{L}(\mathcal{A}_{\mathrm{F}}(\alpha)) = \mathcal{L}(\alpha)$.

They also showed that the follow automaton is a quotient of the position automaton, obtained by identifying positions with the same Follow set. For instance, in the position automaton of Example 3, one can see that $\mathsf{Follow}(\alpha, 1) = \mathsf{Follow}(\alpha, 3) = \{1, 2\}$, and that 1 and 3 are both accepting states. Formally, Ilie and Yu considered the right-invariant equivalence relation $\equiv_{\mathrm{F}}$ defined on the set of states $\mathsf{Pos}_0(\alpha)$, w.r.t. $\mathcal{A}_{\mathrm{POS}}(\alpha)$, by

$$i \equiv_{\mathrm{F}} j \;\Leftrightarrow\; \mathsf{Follow}(\alpha, i) = \mathsf{Follow}(\alpha, j) \text{ and } \varepsilon(i) = \varepsilon(j),$$

and showed that $\mathcal{A}_{\mathrm{F}}(\alpha) \simeq \mathcal{A}_{\mathrm{POS}}(\alpha)/\!\equiv_{\mathrm{F}}$.

We show that $\mathcal{A}_{\mathrm{F}}(\alpha)$ can be directly computed from α, by labelling states not with positions $i \in \mathsf{Pos}_0(\alpha)$, but with their Follow sets and their finality. Let

$$\mathcal{A}_{\mathrm{F}}(\alpha) = \langle \mathsf{F}(\alpha), \Sigma, \delta_{\mathrm{F}}, (\mathsf{Follow}(\alpha, 0), \varepsilon(0)), F_{\mathrm{F}} \rangle,$$

where $\mathsf{F}(\alpha) = \{\, (\mathsf{Follow}(\alpha, i), \varepsilon(i)) \mid i \in \mathsf{Pos}_0(\alpha) \,\} \subseteq 2^{\mathsf{Pos}(\alpha)} \times \{\varepsilon, \emptyset\}$, $F_{\mathrm{F}} = \{\, (S, c) \in \mathsf{F}(\alpha) \mid c = \varepsilon \,\}$ and for $(S, c) \in \mathsf{F}(\alpha)$ and $\sigma \in \Sigma$,

$$\delta_{\mathrm{F}}((S, c), \sigma) = \{\, (\mathsf{Follow}(\alpha, j), \varepsilon(j)) \mid j \in \mathsf{Select}(S, \sigma) \,\}.$$

The transition function δ_{F} is defined as a composition of Select with Follow, instead of Follow with Select as for δ_{POS} (and δ_{MY}). It is necessary to include the finality of a position in the label of the corresponding state, since there might be positions with the same Follow, but different finalities. This can, for instance, be observed in the automaton for $\alpha = a(b^\star c)^\star$. With this definition of $\mathcal{A}_{\mathrm{F}}$ we obtain an alternative proof of the result by Ilie and Yu.

Proposition 7. $\mathcal{A}_{\mathrm{F}}(\alpha) \simeq \mathcal{A}_{\mathrm{POS}}(\alpha)/\!\equiv_{\mathrm{F}}$.

Proof. Consider $\varphi_{\mathrm{F}} : \mathsf{Pos}_0(\alpha)/\!\equiv_{\mathrm{F}} \longrightarrow \mathsf{F}(\alpha)$ defined by $\varphi_{\mathrm{F}}([i]) = (\mathsf{Follow}(\alpha, i), \varepsilon(i))$. By definition, φ_{F} is a bijection and preserves initial as well as final states. Furthermore, for $[i] \in \mathsf{Pos}_0(\alpha)/\!\equiv_{\mathrm{F}}$ and $\sigma \in \Sigma$ we have

$$\begin{aligned}
\varphi_{\mathrm{F}}({}^{\delta_{\mathrm{POS}}}\!/\!_{\equiv_{\mathrm{F}}}([i], \sigma)) &= \varphi_{\mathrm{F}}(\{\, [j] \mid j \in \mathsf{Select}(\mathsf{Follow}(\alpha, i), \sigma) \,\}) \\
&= \{\, \varphi_{\mathrm{F}}([j]) \mid j \in \mathsf{Select}(\mathsf{Follow}(\alpha, i), \sigma) \,\} \\
&= \delta_{\mathrm{F}}((\mathsf{Follow}(\alpha, i), \varepsilon(i)), \sigma) = \delta_{\mathrm{F}}(\varphi_{\mathrm{F}}([i]), \sigma).
\end{aligned}$$

This shows that φ_{F} is an isomorphism. □

3.2 Derivative Based Constructions

Brzozowski [5] defined a DFA equivalent to a regular expression using the notion of derivative. The derivative of $\alpha \in \mathrm{RE}$ w.r.t. $\sigma \in \Sigma$ is $\sigma^{-1}\alpha$, such that $\mathcal{L}(\sigma^{-1}\alpha) = \{\, w \mid \sigma w \in \mathcal{L}(\alpha)\}$. This notion can be extended to words: $\varepsilon^{-1}\alpha = \alpha$ and $(\sigma w)^{-1}\alpha = w^{-1}(\sigma^{-1}\alpha)$. The set of all derivatives of α, $\{\, w^{-1}\alpha \mid w \in \Sigma^\star \}$ may not be finite. For finiteness, Brzozowski considered the quotient of that set modulo some regular expressions equivalences.

The partial derivative automaton $\mathcal{A}_{\mathrm{PD}}(\alpha)$ of a regular expression α was defined independently by Mirkin [15] and Antimirov [2]. Champarnaud and Ziadi stated the equivalence of the two formulations [6], and proved that $\mathcal{A}_{\mathrm{PD}}$ is a quotient of the $\mathcal{A}_{\mathrm{POS}}$ by a right-invariant relation ($\equiv_c$) [7].

The *prefix automaton* $\mathcal{A}_{\mathrm{Pre}}$ was introduced by Yamamoto [19] as a quotient of the $\mathcal{A}_{\varepsilon\text{-T}}$ automaton. Maia et al. [13] characterised the $\mathcal{A}_{\mathrm{Pre}}$ automaton as a solution of a system of left RE equations and express it as a quotient of $\mathcal{A}_{\mathrm{POS}}$ by a left-invariant equivalence relation ($\equiv_\ell$), i.e., a right-invariant relation w.r.t. the reversal of $\mathcal{A}_{\mathrm{POS}}$, cf. Section 5.

4 Automata Based on Pointed Expressions

Next we review two automata constructions, $\mathcal{A}_{\mathrm{MB}}$ and $\mathcal{A}_{\mathrm{MA}}$, that are based on recent approaches of Asperti et al. [3] and by Nipkow and Traytel [16] using pointed expressions. In a pointed regular expression, several positions are selected, and are graphically marked with a *point* corresponding to a letter. Those automata correspond to two different interpretations of a *pointed* expression, i.e., of a given set of positions S: in the first case, given a letter σ one selects which positions from S correspond to that letter and then determines which possible positions can follow; in the second case the set of positions S corresponds to where one can be *after* reading the letter σ. For instance, the pointed regular expression $a(\bullet bb + \bullet aba)^\star \bullet b$ characterises the set of positions $\{2, 4, 7\}$. Intuitively, these are the positions which have been reached after reading some prefix of an input word. Asperti et al. thought that their algorithm "*au point*" computed a DFA isomorphic to $\mathcal{A}_{\mathrm{MY}}(\alpha)$, but Nipkow and Traytel [16] showed that their construction led to a dual automaton and called it *mark before*, $\mathcal{A}_{\mathrm{MB}}$, while $\mathcal{A}_{\mathrm{MY}}$ was isomorphic to a *mark after*, $\mathcal{A}_{\mathrm{MA}}$. Using the notation of the previous section, a transition in $\mathcal{A}_{\mathrm{MA}}$ is a composition of Follow with Select similarly as described in (1), while in $\mathcal{A}_{\mathrm{MB}}$ it will be a composition of Select with Follow. Because of the behaviour of the transition function δ_{MY} of $\mathcal{A}_{\mathrm{MY}}(\alpha)$, Nipkow and Traytel called this construction *mark after* ($\mathcal{A}_{\mathrm{MA}}(\alpha)$).

In this section, we show that the $\mathcal{A}_{\mathrm{MB}}$ is isomorphic to a quotient of the determinisation of $\mathcal{A}_{\mathrm{F}}$, and as a corollary it follows that $\mathcal{A}_{\mathrm{MA}}$ ($\mathcal{A}_{\mathrm{MY}}$) cannot be smaller than $\mathcal{A}_{\mathrm{MB}}$ (as already stated by Nipkow and Traytel). Moreover, we also consider the case, where one restricts pointed regular expressions with only *one* point marking a position. Obviously, the *mark after* automaton of single pointed expressions is related to the position automaton.

4.1 The Automaton $\mathcal{A}_{\mathrm{MB}}$ *Versus* $\mathrm{D}(\mathcal{A}_{\mathrm{F}})$

As mentioned above, Asperti *et al.* introduced the notion of pointed regular expression in order to obtain a compact representation of a set of positions. However, a point was used to mark a position to be visited when reading a letter instead of a position reached after reading the letter, as is the case for $\mathcal{A}_{\mathrm{POS}}$ and $\mathcal{A}_{\mathrm{MA}}$. The resulting construction was called *mark before*, $\mathcal{A}_{\mathrm{MB}}$, by Nipkow and Traytel. In our framework, this means that δ_{MB} is a composition of Follow with Select. Formally, given $\alpha \in \mathrm{RE}$, let

$$\mathcal{A}_{\mathrm{MB}}(\alpha) = \langle Q_{\mathrm{MB}}, \Sigma, \delta_{\mathrm{MB}}, (\mathsf{Follow}(\alpha, 0), \varepsilon(0)), F_{\mathrm{MB}} \rangle,$$

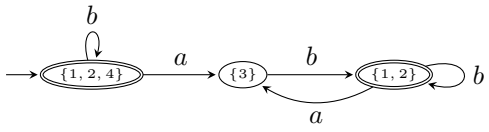
where $Q_{\mathrm{MB}} \subseteq 2^{\mathsf{Pos}(\alpha)} \times \{\emptyset, \varepsilon\}$, and for $(S, c) \in Q_{\mathrm{MB}}$ and $\sigma \in \Sigma$,

$$\delta_{\mathrm{MB}}((S, c), \sigma) = (\mathsf{Follow}(\alpha, \mathsf{Select}(S, \sigma)), \varepsilon(\mathsf{Select}(S, \sigma))),$$

and $F_{\mathrm{MB}} = \{\, (S, c) \mid c = \varepsilon \,\}$. In Q_{MB} we consider only the states that are accessible from the initial state by δ_{MB}.

Proposition 8 ([3, 16]). $\mathcal{L}(\mathcal{A}_{\mathrm{MB}}(\alpha)) = \mathcal{L}(\alpha)$.

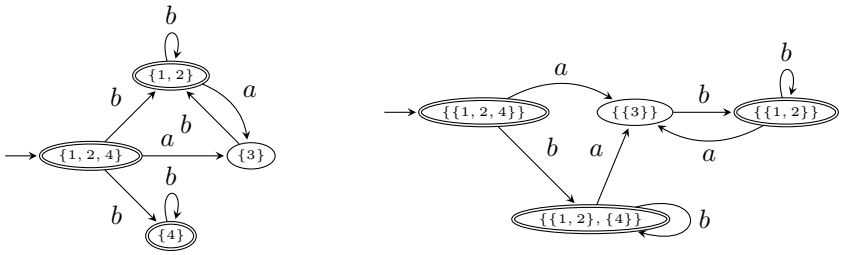
Example 9. Consider again the regular expression α from Example 3. The $\mathcal{A}_{\mathrm{MB}}(\alpha)$ DFA is depicted below.



Note that the first state label is the set $\mathsf{First}(\alpha)$, and one can see that two states are saved when comparing with $\mathcal{A}_{\mathrm{MA}}$, in Example 5. □

One could expect that the $\mathcal{A}_{\mathrm{MB}}$ construction was isomorphic to the determinisation of $\mathcal{A}_{\mathrm{F}}$. But we will see that in general that is not the case. The determinisation of $\mathcal{A}_{\mathrm{F}}$, $\mathrm{D}(\mathcal{A}_{\mathrm{F}}(\alpha)) = \langle Q_{\mathrm{D}(\mathcal{A}_{\mathrm{F}})}, \Sigma, \{(\mathsf{Follow}(\alpha, 0), \varepsilon(0))\}, \delta_{\mathrm{D}(\mathcal{A}_{\mathrm{F}})}, F_{\mathrm{D}(\mathcal{A}_{\mathrm{F}})} \rangle$, can be obtained by the subset construction.

Example 10. Considering again the regular expression α from Example 3, the $\mathcal{A}_{\mathrm{F}}(\alpha)$ and $\mathrm{D}(\mathcal{A}_{\mathrm{F}}(\alpha))$ are respectively:



It is clear that $\mathrm{D}(\mathcal{A}_{\mathrm{F}}(\alpha))$ is not isomorphic to $\mathcal{A}_{\mathrm{MB}}(\alpha)$ (see Example 9). However if one merges the states labeled by $\{(\{1, 2, 4\}, \varepsilon)\}$ and $\{(\{1, 2\}, \varepsilon), (\{4\}, \varepsilon)\}$

in $\mathrm{D}(\mathcal{A}_{\mathrm{F}}(\alpha))$, the DFA $\mathcal{A}_{\mathrm{MB}}(\alpha)$ is obtained. Next we prove that if certain sets of sets in the determinisation of $\mathcal{A}_{\mathrm{F}}$ are flattened the resulting automaton is isomorphic to $\mathcal{A}_{\mathrm{MB}}$. □

Let $\equiv_s$ be the equivalence relation on $2^{\mathsf{F}(\alpha)}$ defined by,

$$I \equiv_s J \;\Leftrightarrow\; \bigcup_{(S,\_)\in I} S = \bigcup_{(S,\_)\in J} S \text{ and } \varepsilon(I) = \varepsilon(J).$$

Proposition 11. $\mathcal{L}\left(\mathrm{D}(\mathcal{A}_{\mathrm{F}}(\alpha))/\equiv_s\right) = \mathcal{L}(\mathrm{D}(\mathcal{A}_{\mathrm{F}}(\alpha)))$.

Proposition 12. $\mathrm{D}(\mathcal{A}_{\mathrm{F}}(\alpha))/\equiv_s \simeq \mathcal{A}_{\mathrm{MB}}(\alpha)$.

Nipkow and Traytel presented a homomorphism from $\mathcal{A}_{\mathrm{MA}}$ to $\mathcal{A}_{\mathrm{MB}}$, showing that $\mathcal{A}_{\mathrm{MA}}$ cannot be smaller than $\mathcal{A}_{\mathrm{MB}}$. The same result is a direct corollary of the above results.

Corollary 13. *Let φ_{F} be defined as in the proof of Proposition 7. Then we have* $\varphi_{\mathrm{F}}\left(\mathrm{D}\left(\mathcal{A}_{\mathrm{POS}}(\alpha)/\equiv_{\mathrm{F}}\right)\right)/\equiv_s = \varphi_{\mathrm{F}}\left(\mathrm{D}(\mathcal{A}_{\mathrm{POS}}(\alpha))/\equiv_{\mathrm{F}}\right)/\equiv_s \simeq \mathcal{A}_{\mathrm{MB}}(\alpha)$.

Example 14. Considering $\mathcal{A}_{\mathrm{MY}}$ ($\mathcal{A}_{\mathrm{MA}}$) from Example 5, we have $\{1\} \equiv_{\mathrm{F}} \{3\}$, since $[1]_{\equiv_{\mathrm{F}}} = [3]_{\equiv_{\mathrm{F}}}$. Furthermore,

$$\varphi_{\mathrm{F}}(\{[0]_{\equiv_{\mathrm{F}}}\}) = \{(\{1,2,4\},\varepsilon)\} \equiv_s \{(\{1,2\},\varepsilon),(\{4\},\varepsilon)\} = \varphi_{\mathrm{F}}(\{[1]_{\equiv_{\mathrm{F}}},[4]_{\equiv_{\mathrm{F}}}\}).$$ □

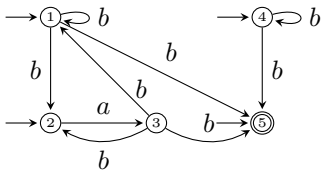
4.2 The Dual Position Automaton

If one considers pointed regular expressions with only one point marking a position to be visited when reading a letter, an NFA, dual of $\mathcal{A}_{\mathrm{POS}}$ ($\mathcal{A}_{\overleftarrow{\mathrm{POS}}}$), can be defined. We show that its determinisation yields $\mathcal{A}_{\mathrm{MB}}$. Given a regular expression α, with $n = |\alpha|_\Sigma = |\,\mathsf{Pos}(\alpha)|$, the set of states of $\mathcal{A}_{\overleftarrow{\mathrm{POS}}}$ is $\mathsf{Pos}(\alpha)$ plus an unique final state $n+1$. The set of initial states is $\mathsf{Follow}(\alpha,0) \cup \varepsilon(\alpha)\{n+1\}$. From a state $i \in \mathsf{Pos}(\alpha)$ reading $\sigma \in \Sigma$ one can move to $\mathsf{Follow}(\alpha,i)$ if $\overline{\sigma_i} = \sigma$. That is, by first selecting $\mathsf{Select}(\{i\},\sigma)$ which is i if $\overline{\sigma_i} = \sigma$, and empty otherwise, and then applying Follow. Moreover, if $\varepsilon(i) = \varepsilon$ there is a transition to $n+1$. Formally,

$$\mathcal{A}_{\overleftarrow{\mathrm{POS}}}(\alpha) = \langle \mathsf{Pos}(\alpha) \cup \{n+1\}, \Sigma, \delta_{\overleftarrow{\mathrm{POS}}}, \mathsf{Follow}(\alpha,0) \cup \varepsilon(\alpha)\{n+1\}, \{n+1\}\rangle,$$

with $\delta_{\overleftarrow{\mathrm{POS}}}(i,\sigma) = \mathsf{Follow}(\alpha, \mathsf{Select}(\{i\},\sigma)) \cup \varepsilon(\mathsf{Select}(\{i\},\sigma))\{n+1\}$. This means that $\delta_{\overleftarrow{\mathrm{POS}}}(i,\sigma) = \mathsf{Follow}(\alpha,i) \cup \varepsilon(i)\{n+1\}$, only if $i \in \mathsf{Pos}(\alpha)$ and $\overline{\sigma_i} = \sigma$, being the empty set otherwise.

Example 15. Considering again the regular expression α from Example 3, the $\mathcal{A}_{\overleftarrow{\mathrm{POS}}}(\alpha)$ is the following:



Observe that for each state of $\mathcal{A}_{\overleftarrow{\mathrm{POS}}}$ all transitions *leaving* it have the same label. This is exactly the opposite of the position automaton $\mathcal{A}_{\mathrm{POS}}$, where for each state all transitions *into it* have the same label.

Proposition 16. $\mathrm{D}(\mathcal{A}_{\overleftarrow{\mathrm{POS}}}(\alpha)) \simeq \mathcal{A}_{\mathrm{MB}}(\alpha)$.

5 Reversals and Automata Constructions

Given a language L the reversal of L, L^{R}, is the language obtained by reversing all the words in L. The reversal of a regular expression α is denoted by α^{R}, and is inductively defined by: $\alpha^{\mathrm{R}} = \alpha$ for $\alpha \in \Sigma \cup \{\varepsilon, \emptyset\}$, $(\alpha+\beta)^{\mathrm{R}} = \beta^{\mathrm{R}}+\alpha^{\mathrm{R}}$, $(\alpha\beta)^{\mathrm{R}} = \beta^{\mathrm{R}}\alpha^{\mathrm{R}}$ and $(\alpha^\star)^{\mathrm{R}} = (\alpha^{\mathrm{R}})^\star$. The reversal α^{R} describes $\mathcal{L}(\alpha)^{\mathrm{R}}$. In the same way, given an automaton $\mathcal{A} = \langle Q, \Sigma, \delta, I, F\rangle$ its reversal is $\mathcal{A}^{\mathrm{R}} = \langle Q, \Sigma, \delta^{\mathrm{R}}, F, I\rangle$, where $\delta^{\mathrm{R}}(q,\sigma) = \{\, p \mid q \in \delta(p,\sigma) \,\}$ and $\mathcal{L}(\mathcal{A}^{\mathrm{R}}) = \mathcal{L}(\mathcal{A})^{\mathrm{R}}$.

Given α, any of the automata constructions in the previous sections can be applied to α^{R}. If one reverses the resulting automaton, an alternative automaton construction for $\mathcal{L}(\alpha)$ is obtained. In this section we establish some relations between the direct constructions and the double reversed ones. We show that $\mathcal{A}_{\mathrm{POS}}(\alpha^{\mathrm{R}})^{\mathrm{R}} \simeq \mathcal{A}_{\overleftarrow{\mathrm{POS}}}$. We also show that determinising any quotient of $\mathcal{A}_{\mathrm{POS}}(\alpha^{\mathrm{R}})^{\mathrm{R}}$ by a right-invariant relation is the same as determinising $\mathcal{A}_{\mathrm{POS}}(\alpha^{\mathrm{R}})^{\mathrm{R}}$ and thus, by Proposition 16, the resulting automata are all isomorphic to $\mathcal{A}_{\mathrm{MB}}$. The same does not hold if one considers quotients by a left-invariant relation, and we illustrate that with the $\mathcal{A}_{\mathrm{Pre}}$ construction.

Our first result on reversals of expressions and automata reads as follows:

Proposition 17. $\mathcal{A}_{\mathrm{POS}}(\alpha^{\mathrm{R}})^{\mathrm{R}} \simeq \mathcal{A}_{\overleftarrow{\mathrm{POS}}}(\alpha)$.

For the position automaton several quotients were presented in Section 3 for which different DFAs could be obtained by determinisation. For the dual construction, $\mathcal{A}_{\overleftarrow{\mathrm{POS}}}$, the determinisation of any quotient by a right-invariant relation is isomorphic to $\mathcal{A}_{\mathrm{MB}}$. The following simple lemma explains the reason.

Lemma 18. *Let A be a trim NFA and consider $\equiv$ a right-invariant relation w.r.t. A. Then,* $\mathrm{D}(A^{\mathrm{R}}/{\equiv}) = \mathrm{D}(A^{\mathrm{R}})$.

This result has direct consequences for all constructions, that can be obtained as a quotient of the position automaton by some right-invariant relation. In particular, we have

Proposition 19. $\mathrm{D}(\mathcal{A}_{\mathrm{POS}}(\alpha^{\mathrm{R}})^{\mathrm{R}}) \simeq \mathrm{D}(\mathcal{A}_{\mathrm{PD}}(\alpha^{\mathrm{R}})^{\mathrm{R}}) \simeq \mathrm{D}(\mathcal{A}_{\mathrm{F}}(\alpha^{\mathrm{R}})^{\mathrm{R}}) \simeq \mathcal{A}_{\mathrm{MB}}(\alpha)$.

Note that Lemma 18 does not hold for left-invariant relations. In particular, one can consider the $\mathcal{A}_{\mathrm{Pre}}$ construction mentioned in Section 3.2.

Proposition 20. *For $\alpha = a^\star + (a+b)a^\star$,* $\mathrm{D}(\mathcal{A}_{\mathrm{Pre}}(\alpha^{\mathrm{R}})^{\mathrm{R}}) \not\simeq \mathrm{D}(\mathcal{A}_{\mathrm{POS}}(\alpha^{\mathrm{R}})^{\mathrm{R}})$.

However, Lemma 18 also implies that if a relation $\equiv$ is a left-invariant equivalence relation w.r.t an NFA A then $\mathrm{D}(A/{\equiv}) = \mathrm{D}(A)$. In particular, the determinisation of $\mathcal{A}_{\mathrm{Pre}}$ is isomorphic to the determinisation of $\mathcal{A}_{\mathrm{POS}}$, i.e., $\mathcal{A}_{\mathrm{MY}}$.

Proposition 21. $\mathrm{D}(\mathcal{A}_{\mathrm{Pre}}(\alpha)) \simeq \mathcal{A}_{\mathrm{MY}}(\alpha)$.

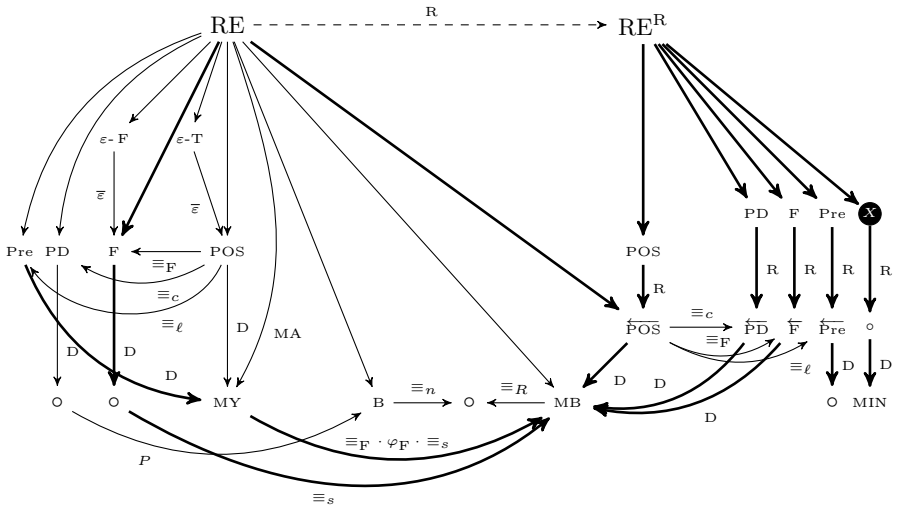


Fig. 1. Taxonomy of conversions from regular expressions to finite automata. Bold arrows correspond to relations studied in this work. Here X may be any construction that yields a DFA.

6 Taxonomy

In Figure 1 the relations between the different automata are graphically represented. The two top nodes correspond to regular expressions. Each other node corresponds to a particular automaton, up to isomorphism, and edges between two nodes represent transformation algorithms, such as epsilon elimination ($\overline{\varepsilon}$) determinisation (D), reversal (R), quotient by some equivalence relation, or a specific construction. Edges in bold correspond to contributions in this paper. Different nodes represent objects for which there is some witness that distinguishes them. The relation between $\mathrm{D}(\mathcal{A}_{\mathrm{PD}}(\alpha))$ and $\mathcal{A}_{\mathrm{B}}(\alpha)$ was obtained by Nipkow and Traytel, and the ones between $\mathcal{A}_{\mathrm{B}}(\alpha)$ and $\mathcal{A}_{\mathrm{MB}}(\alpha)$ were obtained by Asperti et al.. The resulting automaton (between $\mathcal{A}_{\mathrm{B}}(\alpha)$ and $\mathcal{A}_{\mathrm{MB}}(\alpha)$ in the diagram) is the only one for which we do not have witnesses distinguishing it from the others. Brzozowski [4] showed that for a trim NFA $\mathcal{A}$, $\mathrm{D}(\mathcal{A})$ is minimal if $\mathcal{A}^{\mathrm{R}}$ is deterministic. Consequently, one obtains the nice property that, whenever $X(\alpha)$ is a deterministic automaton, for instance $\mathcal{A}_{\mathrm{MB}}$ and $\mathcal{A}_{\mathrm{MA}}$, then $\mathrm{D}(X(\alpha^{\mathrm{R}})^{\mathrm{R}})$ is the minimal DFA for $\mathcal{L}(\alpha)$. Experimental results suggest that $\mathcal{A}_{\mathrm{MB}}(\alpha)$ is never larger than $\mathrm{D}(\mathcal{A}_{\mathrm{PD}}(\alpha))$, so that should be investigated in future work.

References

1. Allauzen, C., Mohri, M.: A unified construction of the Glushkov, follow, and Antimirov automata. In: Kralovic, R., Urzyczyn, P. (eds.) 31st MFCS. LNCS, vol.

4162, pp. 110–121. Springer (2006)
2. Antimirov, V.M.: Partial derivatives of regular expressions and finite automaton constructions. Theoret. Comput. Sci. 155(2), 291–319 (1996)
3. Asperti, A., Coen, C.S., Tassi, E.: Regular expressions, au point. CoRR abs/1010.2604 (2010), http://arxiv.org/abs/1010.2604
4. Brzozowski, J.A.: Canonical regular expressions and minimal state graphs for definite events. In: Mathematical Theory of Automata, pp. 529–561. MRI Symposia Series, Polytechnic Press, Polytechnic Institute of Brooklyn, NY,12 (1962)
5. Brzozowski, J.: Derivatives of regular expressions. J. Association for Computer Machinery (11), 481–494 (1964)
6. Champarnaud, J.M., Ziadi, D.: From Mirkin's prebases to Antimirov's word partial derivatives. Fundam. Inform. 45(3), 195–205 (2001)
7. Champarnaud, J.M., Ziadi, D.: Canonical derivatives, partial derivatives and finite automaton constructions. Theoret. Comput. Sci. 289, 137–163 (2002)
8. Chen, H., Yu, S.: Derivatives of regular expressions and an application. In: Dinneen, M.J., Khoussainov, B., Nies, A. (eds.) Computation, Physics and Beyond, WTCS 2012. LNCS, vol. 7160, pp. 343–356. Springer (2012)
9. Giammarresi, D., Ponty, J.L., Wood, D.: Glushkov and Thompson constructions: A synthesis. HKUST TCSC-98-11, The Department of Science & Engineering, Theoretical Cmputer Science Group, The Hong Kong University of Science and Technology (1998)
10. Glushkov, V.M.: The abstract theory of automata. Russian Math. Surveys 16, 1–53 (1961)
11. Gruber, H., Holzer, M.: From finite automata to regular expressions and back—a summary on descriptional complexity. Internat. J. Found. Comput. Sci. 26(8), 1009–1040 (Dec 2015)
12. Ilie, L., Yu, S.: Follow automata. Inf. Comput. 186(1), 140–162 (2003)
13. Maia, E., Moreira, N., Reis, R.: Prefix and right-partial derivative automata. In: Soskova, M., Mitrana, V. (eds.) 11th CiE. LNCS, vol. 9136, pp. 258–267. Springer (2015)
14. McNaughton, R., Yamada, H.: Regular expressions and state graphs for automata. IEEE Trans. Comput. 9, 39–47 (1960)
15. Mirkin, B.G.: An algorithm for constructing a base in a language of regular expressions. Engineering Cybernetics 5, 51—57 (1966)
16. Nipkow, T., Traytel, D.: Unified decision procedures for regular expression equivalence. In: Klein, G., Gamboa, R. (eds.) 5th ITP. LNCS, vol. 8558, pp. 450–466. Springer (2014)
17. Sakarovitch, J.: Elements of Automata Theory. Cambridge University Press (2009)
18. Thompson, K.: Regular expression search algorithm. Commun. ACM 11(6), 410–422 (1968)
19. Yamamoto, H.: A new finite automaton construction for regular expressions. In: Bensch, S., Freund, R., Otto, F. (eds.) 6th NCMA. books@ocg.at, vol. 304, pp. 249–264. Österreichische Computer Gesellschaft (2014)