Construction of Tree Automata from a Regular Tree Expression

Ludovic Mignot Nadia Ouali Sebti
Djelloul Ziadi*
Laboratoire LITIS - EA 4108, Université de Rouen, Avenue de l'Université 76801
Saint-Étienne-du-Rouvray Cedex, France
{ludovic.mignot, nadia.ouali-sebti, djelloul.ziadi}@univ-rouen.fr

There exist several methods of computing an automaton recognizing the language denoted by a given regular expression: non exhaustively. The *position automaton* $\mathscr{P}$ due to Glushkov, the *c-continuation automaton* $\mathscr{C}$ due to Champarnaud and Ziadi, the *follow automaton* $\mathscr{F}$ due to Ilie and Yu and the equation automaton $\mathscr{E}$ due to Antimirov. It has been shown that $\mathscr{P}$ and $\mathscr{C}$ are isomorphic and that $\mathscr{E}$ (resp. $\mathscr{F}$) is a quotient of $\mathscr{C}$ (resp. of $\mathscr{P}$). In this paper, we define from a given regular tree expression the k-position tree automaton $\mathscr{P}$ and the follow tree automaton $\mathscr{F}$. Using the definition of the equation tree automaton $\mathscr{E}$ of Kuske and Meinecke and our previously defined k-C-continuation tree automaton $\mathscr{C}$, we show that the previous morphic relations are still valid on regular tree expressions.

1 Introduction

Regular expressions are used in numerous domains of applications in computer science. They are an easy and compact way to represent potentially infinite regular languages, that are well-studied objects leading to efficient decision problems. The first approach of the computation of an automaton from regular expression is to determine particular properties over the syntactic structure of the regular expression E. Glushkov [7] proposed the computation of an automaton with $(n+1)$-states. Ilie and Yu showed in [8] how to reduce it by merging similar states. Another method is to compute the transition function of the automaton as follows. Basically, it is a computation that tries to determine what words w' can be accepted after reading a prefix w. The first author that introduced such a process is Brzozowski [2]. He showed how to compute a regular expression denoting $w^{-1}(L(\mathrm{E}))$ from the expression E: this expression, denoted by $d_w(\mathrm{E})$, is called the *derivative* of E with respect to w. Furthermore, the set of dissimilar derivatives, combined with reduction according to associativity, commutativity and idempotence of the sum, is finite and can lead to the computation of a deterministic finite automaton. Antimirov [1] computed the partial derivatives. These so-called derived terms produce the *equation automaton*. Finally, by deriving expressions after having them indexed, Champarnaud and Ziadi [4] computed the *c-continuation automaton*. The different morphic links between these four automata have been studied too: Ilie and Yu showed that the follow automaton is a quotient of the position automaton; Champarnaud and Ziadi proved that the position automaton and the c-continuation automaton are isomorphic and that the equation automaton is a quotient of the position automaton. Finally, using a join of the two previously defined quotients, Garcia *et al.* presented in [6] an automaton that is smaller than both the follow and the equation automata.

In this paper, we recall the study of these already known morphic links between different tree automata. We recall two tree automata constructions, *the k-position automaton* [14] and *the follow automaton* [14], and we recall their morphic links with two other already known automata constructions,

---

[*] D. Ziadi was supported by the MESRS - Algeria under Project 8/U03/7015.

Submitted to:
TTATT 2015

the *equation automaton* of Kuske and Meinecke [10] and our *k-C-continuation automaton* [12, 13, 14]. Notice that a position automaton and a reduced automaton have already been defined in [11]. However, they are not isomorphic with the automata we define in this paper. This study is motivated by the development of a library of functions for handling rational kernels [5] in the case of trees. The first problem consists in converting a regular tree expression into a tree automaton. In Section 2, we recall the definitions of *k-position automaton*, *follow automaton*, *equation automaton* and of the *k-C-continuation automaton*; we also present the morphic links between these four automata in this section. It is proved that there is no morphic link between the follow automaton and the equation automaton. Moreover, we recall the computation of the Garcia *et al.* equivalence leading to a smaller automaton in Section 3. Then, in Section 4 we give the algorithms and the complexity of the computation of different tree automata. Finally, the different results described in this paper are given in the conclusion.

2 Tree Automata from Regular Expressions

In the following we use the definitions of a *ranked alphabet*, *a tree*, *a finite tree automaton*, *a tree substitution*, a *c-product*, an *iterated c-product*, a *c-closure*, a *regular tree expression* and its denoted language, a set of *positions*, a quotient of a tree automaton and a mapping h defined in [12, 13, 14]. A regular expression E is *linear* if every symbol of rank greater than 1 appears at most once in E.

In this section, we show how to compute from a regular expression E several tree automata accepting $[\![\mathrm{E}]\!]$: we introduce two new constructions, the k-position automaton and the follow automaton of E, and then we recall two already-known constructions, the equation [10] and the k-C-continuation [12] automata. Regular languages defined over the ranked alphabet Σ are exactly the languages denoted by a regular expression on Σ. In what follows we only consider expressions without 0 or reduced to 0. The set of symbols in Σ that appear in an expression F is denoted by Σ_F.

2.1 The k-Position Tree Automaton

In this section, we show how to compute the k-position automaton of an expression E, recognizing $[\![E]\!]$. This is an extension of the well-known position automaton [7] for regular word expressions. In what follows, for any two trees s and t, we denote by $s \preccurlyeq t$ the relation "s is a subtree of t". Let $t = f(t_1, \dots, t_n)$ be a tree. We denote by $\mathrm{root}(t)$ the root of t, by k-child(t) the label of the k^{th} child of f in t, that is the root of t_k if it exists, and by $\mathrm{Leaves}(t)$ the set of the leaves of t, *i.e.* $\{s \in \Sigma_0 \mid s \preccurlyeq t\}$. Let E be a regular expression, $1 \leq k \leq m$ be two integers and f be a symbol in Σ_m. The set First(E) is the subset of Σ defined by $\{\mathrm{root}(t) \in \Sigma \mid t \in [\![\mathrm{E}]\!]\}$; The set Follow(E$,f,k$) is the subset of Σ defined by $\{g \in \Sigma \mid \exists t \in [\![\mathrm{E}]\!], \exists s \preccurlyeq t, \mathrm{root}(s) = f, k\text{-child(s)} = g\}$; The set Last(E) is the subset of Σ_0 defined by $\mathrm{Last(E)} = \bigcup_{t \in [\![\mathrm{E}]\!]} \mathrm{Leaves}(t)$. Let us first show that the position functions First and Follow are inductively computable.

Let E be linear. The set First(E) can be computed as follows:

$$\mathrm{First}(0) = \emptyset,\ \mathrm{First}(a) = \{a\},\ \mathrm{First}(f(\mathrm{E}_1, \cdots, \mathrm{E}_m)) = \{f\},$$
$$\mathrm{First}(\mathrm{E}_1 + \mathrm{E}_2) = \mathrm{First}(\mathrm{E}_1) \cup \mathrm{First}(\mathrm{E}_2),\ \mathrm{First}(\mathrm{E}_1{}^{*_c}) = \mathrm{First}(\mathrm{E}_1) \cup \{c\},$$
$$\mathrm{First}(\mathrm{E}_1 \cdot_c \mathrm{E}_2) = \begin{cases} (\mathrm{First}(\mathrm{E}_1) \setminus \{c\}) \cup \mathrm{First}(\mathrm{E}_2) & \text{if } c \in [\![\mathrm{E}_1]\!], \\ \mathrm{First}(\mathrm{E}_1) & \text{otherwise.} \end{cases}$$

Let $1 \leq k \leq m$ be two integers and f be a symbol in Σ_m. The set of symbols Follow(E$,f,k$) can be computed inductively as follows:

$$\mathrm{Follow}(0, f, k) = \mathrm{Follow}(a, f, k) = \emptyset,$$

$$\mathrm{Follow}(g(\mathrm{E}_1,\ldots,\mathrm{E}_n),f,k)=\begin{cases}\mathrm{First}(\mathrm{E}_k) & \text{if } f=g,\\ \mathrm{Follow}(\mathrm{E}_l,f,k) & \text{if } \exists l \mid f\in\Sigma_{E_l},\\ \emptyset & \text{otherwise .}\end{cases}$$

$$\mathrm{Follow}(\mathrm{E}_1+\mathrm{E}_2,f,k)=\begin{cases}\mathrm{Follow}(\mathrm{E}_1,f,k) & \text{if } f\in\Sigma_{\mathrm{E}_1},\\ \mathrm{Follow}(\mathrm{E}_2,f,k) & \text{if } f\in\Sigma_{\mathrm{E}_2},\\ \emptyset & \text{otherwise .}\end{cases}$$

$$\mathrm{Follow}(\mathrm{E}_1\cdot_c\mathrm{E}_2,f,k)=\begin{cases}(\mathrm{Follow}(\mathrm{E}_1,f,k)\setminus\{c\})\cup\mathrm{First}(\mathrm{E}_2) & \text{if } c\in\mathrm{Follow}(\mathrm{E}_1,f,k),\\ \mathrm{Follow}(\mathrm{E}_1,f,k) & \text{if } f\in\Sigma_{\mathrm{E}_1}\wedge c\notin\mathrm{Follow}(\mathrm{E}_1,f,k),\\ \mathrm{Follow}(\mathrm{E}_2,f,k) & \text{if } f\in\Sigma_{\mathrm{E}_2}\wedge c\in\mathrm{Last}(\mathrm{E}_1),\\ \emptyset & \text{otherwise,}\end{cases}$$

$$\mathrm{Follow}(\mathrm{E}_1^{*_c},f,k)=\begin{cases}\mathrm{Follow}(\mathrm{E}_1,f,k)\cup\mathrm{First}(\mathrm{E}_1) & \text{if } c\in\mathrm{Follow}(\mathrm{E}_1,f,k),\\ \mathrm{Follow}(\mathrm{E}_1,f,k) & \text{otherwise,}\end{cases}$$

The two functions First and Follow are sufficient to compute the *k-position tree automaton* of E. The *k-position automaton* $\mathscr{P}_\mathrm{E}$ is the automaton (Q,Σ,Q_T,Δ) defined by

$$Q=\{f^k \mid f\in\Sigma_m\wedge 1\leq k\leq m\}\cup\{\varepsilon^1\} \text{ with } \varepsilon^1 \text{ a new symbol not in } \Sigma,\ Q_T=\{\varepsilon^1\}$$
$$\Delta=\{(f^k,g,g^1,\ldots,g^n)\mid f\in\Sigma_m\wedge k\leq m\wedge g\in\Sigma_n\wedge g\in\mathrm{Follow}(\mathrm{E},f,k)\}$$
$$\cup\{(\varepsilon^1,f,f^1,\ldots,f^m)\mid f\in\Sigma_m\wedge f\in\mathrm{First}(\mathrm{E})\}$$

Theorem 1. *If* E *is linear, then* $\mathscr{L}(\mathscr{P}_\mathrm{E})=[\![\mathrm{E}]\!]$.

2.2 The Follow Tree Automaton

In this section, we define the follow tree automaton which is a generalisation of the Follow automaton introduced by L. Ilie and S. Yu in [8] and that it is a quotient of the k-position automaton. Notice that in this automaton, states are no longer positions, but sets of positions and that we extend the definition of the function Follow to the position ε^1 by $\mathrm{Follow}(E,\varepsilon^1,1)=\mathrm{First}(E)$.

Definition 1. *Let* E *be linear. The* Follow Automaton *of* E *is the tree automaton* $\mathscr{F}_\mathrm{E}=(Q,\Sigma,Q_T,\Delta)$ *defined as follows:*

$$Q=\{\mathrm{First(E)}\}\cup\bigcup_{f\in\Sigma_{Em}}\{\mathrm{Follow}(\mathrm{E},f,k)\mid 1\leq k\leq m\},\ Q_T=\{\mathrm{First(E)}\}$$
$$\Delta=\{(\mathrm{Follow}(\mathrm{E},g,l),f,\mathrm{Follow}(\mathrm{E},f,1),\ldots,\mathrm{Follow}(\mathrm{E},f,m)\mid f\in\Sigma_{Em}$$
$$\wedge f\in\mathrm{Follow}(\mathrm{E},g,l)\wedge g\in\Sigma_n\wedge l\leq n\}\cup\{(I,c)\mid c\in I\wedge c\in\Sigma_0\}$$

Let us show that $\mathscr{F}_\mathrm{E}$ is a quotient of $\mathscr{P}_\mathrm{E}$ w.r.t. a similarity relation ; since this kind of quotient preserves the language, this method is consequently a proof of the fact that the language denoted by E is recognized by $\mathscr{F}_\mathrm{E}$.

A *similarity relation* over an automaton $A=(Q,\Sigma,Q_T,\Delta)$ is an equivalence relation $\sim$ over Q such that for any two states q and q' in Q: $q\sim q'\Rightarrow\forall f\in\Sigma_n$, $\forall(q_1,\ldots,q_n)\in Q^n$, $(q,f,q_1,\ldots,q_n)\in\Delta\Leftrightarrow(q',f,q_1,\ldots,q_n)\in\Delta$. In other words, two similar states admit the same predecessors w.r.t. any symbol.

Proposition 1. *Let* $\mathscr{A}$ *be an automaton and* $\sim$ *be a similarity relation over* $\mathscr{A}$*. Then* $\mathscr{L}(\mathscr{A}_{/\sim})=\mathscr{L}(\mathscr{A})$.

Proposition 2. *Let E be linear. The relation* $\sim_\mathscr{F}$ *is the largest similarity relation over* $\mathscr{P}_E$.

Proposition 3. *Let* E *be linear. The finite tree automaton* $\mathscr{P}_\mathrm{E}/\sim_\mathscr{F}$ *is isomorphic to* $\mathscr{F}_\mathrm{E}$.

Theorem 2. *Let* E *be linear. Then* $\mathscr{L}(\mathscr{F}_\mathrm{E})=[\![\mathrm{E}]\!]$.

2.3 The Equation Tree Automaton

In [10], Kuske and Meinecke extend the notion of word partial derivatives [1] to tree partial derivatives in order to compute from E a tree automaton recognizing $[\![\mathrm{E}]\!]$. Due to the notion of ranked alphabet, partial derivatives are no longer sets of expressions, but sets of tuples of expressions.

Let $\mathscr{N} = (\mathrm{E}_1, \ldots, \mathrm{E}_n)$ be a tuple of regular expressions, F and G be some regular expressions and $c \in \Sigma_0$. Then $\mathscr{N} \cdot_c \mathrm{F}$ is the tuple $(\mathrm{E}_1 \cdot_c \mathrm{F}, \ldots, \mathrm{E}_n \cdot_c \mathrm{F})$. For a set $\mathscr{S}$ of tuples of regular expressions, $\mathscr{S} \cdot_c \mathrm{F}$ is the set $\mathscr{S} \cdot_c \mathrm{F} = \{\mathscr{N} \cdot_c \mathrm{F} \mid \mathscr{N} \in \mathscr{S}\}$. Finally, $\mathrm{SET}(\mathscr{N}) = \{\mathrm{E}_1, \cdots, \mathrm{E}_m\}$ and $\mathrm{SET}(\mathscr{S}) = \bigcup_{\mathscr{N} \in \mathscr{S}} \mathrm{SET}(\mathscr{N})$. Let f be a symbol in $\Sigma_{>0}$. The set $f^{-1}(\mathrm{E})$ of tuples of regular expressions is defined as follows:

$$f^{-1}(0) = \emptyset, f^{-1}(F+G) = f^{-1}(\mathrm{F}) \cup f^{-1}(\mathrm{G}), f^{-1}(\mathrm{F}^{*_c}) = f^{-1}(\mathrm{F}) \cdot_c \mathrm{F}^{*_c},$$

$$f^{-1}(g(\mathrm{E}_1, \cdots, \mathrm{E}_n)) = \begin{cases} \{(\mathrm{E}_1, \cdots, \mathrm{E}_n)\} & \text{if } f = g, \\ \emptyset & \text{otherwise,} \end{cases} \quad f^{-1}(\mathrm{F} \cdot_c \mathrm{G}) = \begin{cases} f^{-1}(\mathrm{F}) \cdot_c \mathrm{G} & \text{if } c \notin \llbracket \mathrm{F} \rrbracket \\ f^{-1}(\mathrm{F}) \cdot_c \mathrm{G} \cup f^{-1}(\mathrm{G}) & \text{otherwise.} \end{cases}$$

The function f^{-1} is extended to any set S of regular expressions by $f^{-1}(S) = \bigcup_{\mathrm{E} \in S} f^{-1}(\mathrm{E})$.
The *partial derivative* of E w.r.t. a word $w \in \Sigma_{\geq 1}^*$, denoted by $\partial_w(\mathrm{E})$, is the set of regular expressions inductively defined by: $\partial_w(\mathrm{E}) = \begin{cases} \{\mathrm{E}\} & \text{if } w = \varepsilon, \\ \mathrm{SET}(f^{-1}(\partial_u(\mathrm{E}))) & \text{if } w = uf, f \in \Sigma_{\geq 1}, u \in \Sigma_{\geq 1}^*, f^{-1}(\partial_u(\mathrm{E})) \neq \emptyset, \\ \{0\} & \text{if } w = uf, f \in \Sigma_{\geq 1}, u \in \Sigma_{\geq 1}^*, f^{-1}(\partial_u(\mathrm{E})) = \emptyset. \end{cases}$

The *Equation Automaton* of E is the tree automaton $\mathscr{A}_{\mathrm{E}} = (Q, \Sigma, Q_T, \Delta)$ defined by $Q = \{\partial_w(\mathrm{E}) \mid w \in \Sigma_{\geq 1}^*\}$, $Q_T = \{\mathrm{E}\}$, and

$$\begin{aligned} \Delta = \; & \{(\mathrm{F}, f, \mathrm{G}_1, \ldots, \mathrm{G}_m) \mid \mathrm{F} \in Q, f \in \Sigma_m, m \geq 1, (\mathrm{G}_1, \ldots, \mathrm{G}_m) \in f^{-1}(\mathrm{F})\} \\ & \cup \{(\mathrm{F}, c) \mid \mathrm{F} \in Q \wedge c \in (\llbracket \mathrm{F} \rrbracket \cap \Sigma_0)\} \end{aligned}$$

2.4 The *k*-C-Continuation Tree Automaton

In [10], Kuske and Meinecke show how to efficiently compute the equation tree automaton of a regular expression *via* an extension of Champarnaud and Ziadi's C-Continuation [3, 4, 9]. In [12, 13, 14], we show how to inductively compute them. In this section, we prove that this automaton is isomorphic to the *k*-position tree automaton and we consider the following quotient: $0 \cdot_c \mathrm{E} = 0$. As we consider only regular expressions without 0 or reduced to 0 then if after the computation of *k*-C-Continuation we obtain expression of the form $0 \cdot_c \mathrm{E}$ we reduce it to 0.

Definition 2 ([12, 13, 14]). *Let* $\mathrm{E} \neq 0$ *be linear. Let k and m be two integers such that* $1 \leq k \leq m$. *Let f be in* $(\Sigma_{\mathrm{E}} \cap \Sigma_m)$. *The k*-C-continuation $C_{f^k}(\mathrm{E})$ *of f in* E *is the regular expression defined by:*

$$C_{f^k}(g(\mathrm{E}_1, \cdots, \mathrm{E}_m)) = \begin{cases} \mathrm{E}_k & \textit{if } f = g \\ C_{f^k}(\mathrm{E}_j) & \textit{if } f \in \Sigma_{\mathrm{E}_j} \end{cases} \quad C_{f^k}(\mathrm{E}_1 + \mathrm{E}_2) = \begin{cases} C_{f^k}(\mathrm{E}_1) & \textit{if } f \in \Sigma_{\mathrm{E}_1} \\ C_{f^k}(\mathrm{E}_2) & \textit{if } f \in \Sigma_{\mathrm{E}_2} \end{cases}$$

$$C_{f^k}(\mathrm{E}_1 \cdot_c \mathrm{E}_2) = \begin{cases} C_{f^k}(\mathrm{E}_1) \cdot_c \mathrm{E}_2 & \textit{if } f \in \Sigma_{\mathrm{E}_1} \\ C_{f^k}(\mathrm{E}_2) & \textit{if } f \in \Sigma_{\mathrm{E}_2} \\ & \textit{and } c \in \mathrm{Last}(E_1) \\ 0 & \textit{otherwise} \end{cases} \quad C_{f^k}(\mathrm{E}_1{}^{*_c}) = C_{f^k}(\mathrm{E}_1) \cdot_c \mathrm{E}_1{}^{*_c}.$$

By convention, we set $C_{\varepsilon^1}(\mathrm{E}) = \mathrm{E}$.

Lemma 1. *Let* E *be a regular expression without occurrences of* 0 *or reduced to* 0. *Then,* $C_{f^k}(\mathrm{E})$ *is a regular expression without occurrences of* 0 *or reduced to* 0.

Let us now show how to compute the *k*-C-Continuation tree automaton.

Definition 3 ([12, 13, 14]). *Let* $\mathrm{E} \neq 0$ *be linear. The automaton* $\mathscr{C}_{\mathrm{E}} = (Q_{\mathscr{C}}, \Sigma_E, \{C_{\varepsilon^1}(\mathrm{E})\}, \Delta_{\mathscr{C}})$ *is defined:*

$$\begin{aligned} Q_{\mathscr{C}} &= \{(f^k, C_{f^k}(\mathrm{E})) \mid f \in \Sigma_m, 1 \leq k \leq m\} \cup \{(\varepsilon^1, C_{\varepsilon^1}(\mathrm{E}))\}, \\ \Delta_{\mathscr{C}} &= \{((x, C_x(\mathrm{E})), g, ((g^1, C_{g^1}(\mathrm{E})), \ldots, (g^m, C_{g^m}(\mathrm{E})))) \mid g \in \Sigma_{Em}, \\ & \quad m \geq 1, (C_{g^1}(\mathrm{E}), \ldots, C_{g^m}(\mathrm{E})) \in g^{-1}(C_x(\mathrm{E}))\} \cup \{((x, C_x(\mathrm{E})), c) \mid, c \in \llbracket C_x(\mathrm{E}) \rrbracket \cap \Sigma_0\} \end{aligned}$$

Theorem 3 ([12, 13, 14]). *The automaton* $\mathscr{C}_{\mathrm{E}}$ *accepts* $\llbracket \mathrm{E} \rrbracket$.

Let $\sim_e$ be the equivalence relation over the set of states of $\mathscr{C}_{\mathrm{E}}$ defined for any two states $(f_j^k, C_{f_j^k}(\overline{\mathrm{E}}))$ and $(g_i^p, C_{g_i^p}(\overline{\mathrm{E}}))$ by $(f_j^k, C_{f_j^k}(\overline{\mathrm{E}})) \sim_e (g_i^p, C_{g_i^p}(\overline{\mathrm{E}})) \Leftrightarrow h(C_{f_j^k}(\overline{\mathrm{E}})) = h(C_{g_i^p}(\overline{\mathrm{E}}))$.

Proposition 4 ([12, 13, 14])**.** *The automaton* $\mathscr{C}_{\mathrm{E}}/_{\sim_e}$ *is isomorphic to* $\mathscr{A}_{\mathrm{E}}$.

Proposition 5. *The Follow tree automaton and the Equation Tree Automaton are incomparable though they are derived from two isomorphic automata,* i.e. *neither is a quotient of the other.*

3 A smaller automaton

In [6] P. García *et al.* proposed an algorithm to obtain an automaton from a word regular expression. Their method is based on the computation of both the partial derivatives automaton and the follow automaton. They join two relations, the first relation is over the states of the word follow automaton and the second relation is over the word c-continuations automaton, in one relation denoted by $\equiv_V$. What we propose is to extend the relation $\equiv_V$ to the case of trees as follows:

$$C_{f_j^k}(\overline{\mathrm{E}}) \equiv_V C_{g_i^p}(\overline{\mathrm{E}}) \Leftrightarrow \begin{cases} (\exists C_{h_m^l}(\overline{\mathrm{E}}) \sim_{\mathscr{F}} C_{f_j^k}(\overline{\mathrm{E}}) \mid C_{h_m^l}(\overline{\mathrm{E}}) \sim_e C_{g_i^p}(\overline{\mathrm{E}})) \\ \vee\, (\exists C_{h_m^l}(\overline{\mathrm{E}}) \sim_{\mathscr{F}} C_{g_i^p}(\overline{\mathrm{E}})) \mid C_{h_m^l}(\overline{\mathrm{E}}) \sim_e C_{f_j^k}(\overline{\mathrm{E}})) \end{cases}$$

The idea is to define the follow relation $\sim_{\mathscr{F}}$ over the states of the k-c-continuation automaton $\mathscr{C}_{\mathrm{E}}$ as follows: $C_{f_j^k}(\overline{\mathrm{E}}) \sim_{\mathscr{F}} C_{g_i^p}(\overline{\mathrm{E}}) \Leftrightarrow \mathrm{Follow}(C_{f_j^k}(\overline{\mathrm{E}}), f_j, k) = \mathrm{Follow}(C_{g_i^p}(\overline{\mathrm{E}}), g_i, p)$ such that we keep all the equivalent k-c-continuations in the merged states. The obtained automaton is denoted by $\mathscr{C}_{\mathrm{E}}/_{\sim_{\mathscr{F}}}$. Then apply the relation $\sim_e$ (apply the mapping h) over the states of the automaton $\mathscr{C}_{\mathrm{E}}/_{\sim_{\mathscr{F}}}$ and merge the states which have at least one expression in common.

4 Complexity of the computation of the tree automata

In [10], Kuske and Meinecke extend the algorithm based on the notion of word partial derivatives [1] to tree partial derivatives in order to compute from a regular expression E a tree automaton recognizing $[\![\mathrm{E}]\!]$ with a complexity $O(|\mathrm{E}|^2)$. Laugerotte et al. proposed an algorithm for the computation of the position tree automaton and the reduced tree automaton with an $O(||\mathrm{E}|| \cdot |\mathrm{E}|)$ space and time complexity in [11] with $||\mathrm{E}||$ is the alphabetic width of E and $|\mathrm{E}|$ is its size. In [12, 13] Mignot et al. gave an efficient algorithm for the computation of the equation automaton using the k-c-continuations with an $O(||\mathrm{E}|| \cdot |Q|)$ space and time complexity where $|Q|$ is the set of k-c-continuations of E. The algorithm proposed in [11] for the computation of the function Follow can be used in different constructions such us the equation automaton [10], *k-c-continuation automaton* [12, 13, 14] and Follow Automaton [14].

5 Conclusion

In this paper we define and recall different constructions of tree automata from a regular expression. The different automata and their relations (quotient, isomorphism) defined in this paper are represented in Figure 1, where our extension of García *et al.* construction is denoted by $\equiv_V$-NFA. We have shown that the k-position automaton and the k-c-continuations automaton are isomorphic, and that both the equation automaton and the follow autolaton are different quotients of the k-position automaton.

Looking for reductions of the set of states, we applied the algorithm by García *et al.* [6] which allowed us to compute an automaton the size of which is bounded above by the size of the smaller of the follow and the equation automata.

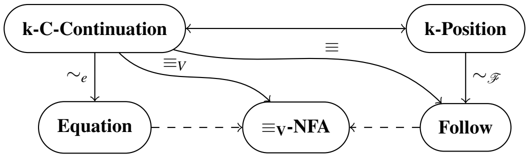

Figure 1: Relations between the automata.

References

[1] Valentin M. Antimirov (1996): *Partial Derivatives of Regular Expressions and Finite Automaton Constructions*. Theor. Comput. Sci. 155(2), pp. 291–319. Available at `http://dx.doi.org/10.1016/0304-3975(95)00182-4`.

[2] Janusz A. Brzozowski (1964): *Derivatives of Regular Expressions*. J. ACM 11(4), pp. 481–494. Available at `http://doi.acm.org/10.1145/321239.321249`.

[3] Jean-Marc Champarnaud & Djelloul Ziadi (2001): *From C-Continuations to New Quadratic Algorithms for Automaton Synthesis*. IJAC 11(6), pp. 707–736. Available at `http://dx.doi.org/10.1142/S0218196701000772`.

[4] Jean-Marc Champarnaud & Djelloul Ziadi (2002): *Canonical derivatives, partial derivatives and finite automaton constructions*. Theor. Comput. Sci. 289(1), pp. 137–163. Available at `http://dx.doi.org/10.1016/S0304-3975(01)00267-5`.

[5] Corinna Cortes, Patrick Haffner & Mehryar Mohri (2004): *Rational Kernels: Theory and Algorithms*. Journal of Machine Learning Research 5, pp. 1035–1062. Available at `http://www.ai.mit.edu/projects/jmlr/papers/volume5/cortes04a/cortes04a.pdf`.

[6] Pedro García, Damián López, José Ruiz & Gloria Inés Alvarez (2011): *From regular expressions to smaller NFAs*. Theor. Comput. Sci. 412(41), pp. 5802–5807. Available at `http://dx.doi.org/10.1016/j.tcs.2011.05.058`.

[7] V.-M. Glushkov (1961): *The abstract theory of automata*. Russian Mathematical Surveys 16, pp. 1–53.

[8] Lucian Ilie & Sheng Yu (2003): *Follow automata*. Inf. Comput. 186(1), pp. 140–162. Available at `http://dx.doi.org/10.1016/S0890-5401(03)00090-7`.

[9] Ahmed Khorsi, Faissal Ouardi & Djelloul Ziadi (2008): *Fast equation automaton computation*. J. Discrete Algorithms 6(3), pp. 433–448. Available at `http://dx.doi.org/10.1016/j.jda.2007.10.003`.

[10] Dietrich Kuske & Ingmar Meinecke (2011): *Construction of tree automata from regular expressions*. RAIRO - Theor. Inf. and Applic. 45(3), pp. 347–370. Available at `http://dx.doi.org/10.1051/ita/2011107`.

[11] Éric Laugerotte, Nadia Ouali Sebti & Djelloul Ziadi (2013): *From Regular Tree Expression to Position Tree Automaton*. In Adrian Horia Dediu, Carlos Martín-Vide & Bianca Truthe, editors: LATA, Lecture Notes in Computer Science 7810, Springer, pp. 395–406. Available at `http://dx.doi.org/10.1007/978-3-642-37064-9_35`.

[12] Ludovic Mignot, Nadia Ouali Sebti & Djelloul Ziadi (2014): *An Efficient Algorithm for the Equation Tree Automaton via the k-C-Continuations*. In Arnold Beckmann, Erzsébet Csuhaj-Varjú & Klaus Meer, editors: CiE, Lecture Notes in Computer Science 8493, Springer, pp. 303–313. Available at `http://dx.doi.org/10.1007/978-3-319-08019-2_31`.

[13] Ludovic Mignot, Nadia Ouali Sebti & Djelloul Ziadi (2014): *An Efficient Algorithm for the Equation Tree Automaton via the $k$-C-Continuations*. CoRR abs/1401.5951. Available at `http://arxiv.org/abs/1401.5951`.

[14] Ludovic Mignot, Nadia Ouali Sebti & Djelloul Ziadi (2014): *$k$-Position, Follow, Equation and $k$-C-Continuation Tree Automata Constructions*. In Zoltán Ésik & Zoltán Fülöp, editors: AFL, EPTCS 151, pp. 327–341. Available at `http://dx.doi.org/10.4204/EPTCS.151.23`.