K-Position, Follow, Equation and K-C-Continuation Tree Automata Constructions

Ludovic Mignot
Laboratoire LITIS - EA 4108 Université de Rouen, Avenue de l'Université 76801 Saint-Étienne-du-Rouvray Cedex, France
ludovic.mignot@univ-rouen.fr

Nadia Ouali Sebti
Laboratoire LITIS - EA 4108 Université de Rouen, Avenue de l'Université 76801 Saint-Étienne-du-Rouvray Cedex, France
nadia.ouali-sebti@univ-rouen.fr

Djelloul Ziadi*
Laboratoire LITIS - EA 4108 Université de Rouen, Avenue de l'Université 76801 Saint-Étienne-du-Rouvray Cedex, France
djelloul.ziadi@univ-rouen.fr

There exist several methods of computing an automaton recognizing the language denoted by a given regular expression: In the case of words, 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 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. Among them, the membership test, that is to determine whether or not a given word belongs to a language. Given a regular expression E with n symbols and a word w, to determine whether w is in the language denoted by E can be polynomially performed (with respect to n) *via* the computation of a finite state machine, called an automaton, that can be seen as a symbol-labelled graph with initial and final states. There exist several methods to compute such an automaton.

The first approach is to determine particular properties over the syntactic structure of the regular expression E. Glushkov [8] proposed the computation of four position functions Null, First, Last, and Follow, which once computed, lead to the computation of a $(n+1)$- state automaton. Ilie and Yu showed in [9] how to reduce it by merging similar states. Another method is to compute the transition function of the automaton as follows: associating a regular expression with a state s, any path labelled by a word w brings the automaton from the state s into a finite set of states $S' = \{s'_1, \ldots, s'_k\}$ such that these states denote the quotient $w^{-1}(L(s))$ of the language $L(s)$ by w, that contains the word w' such that ww' belongs to $L(s)$. 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

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

Z. Ésik and Z. Fülöp (Eds.): Automata and Formal Languages 2014 (AFL 2014)
EPTCS 151, 2014, pp. 327–341, doi:10.4204/EPTCS.151.23

to compute a regular expression denoting $w^{-1}(L(E))$ from the expression E: this expression, denoted by $d_w(E)$, 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] extended this method to the computation of partial derivatives, that are no longer expressions but sets of expressions. 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 [7] an automaton that is smaller than both the follow and the equation automata.

In this paper, we extend the study of these morphic links to different computations of tree automata. We define two new tree automata constructions, *the k-position automaton* and *the follow automaton*, and we study their morphic links with two other already known automata constructions, the *equation automaton* of Kuske and Meinecke [11] and our *k-C-continuation automaton* [14,15]. Notice that a position automaton and a reduced automaton have already been defined in [12]. 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 [6] in the case of trees. The first problem consists in converting a regular tree expression into a tree transducer. Section 2 recalls basic definitions and properties of regular tree languages and regular tree expressions. In Section 3, we define two new automata computations, *the k-position automaton* and *the follow automaton* and recall the definition of the *equation automaton* and of the *k-C-continuation automaton*; we also present the morphic links between these four methods in this section. Section 4 is devoted to the comparison of the follow automaton and of the equation automaton; it is proved that there are no morphic link between them. Moreover, we extend the computation of the Garcia *et al.* equivalence leading to a smaller automaton in this section.

2 Preliminaries

Let (Σ, ar) be *a ranked alphabet*, where Σ is a finite set and ar represents the *rank* of Σ which is a mapping from Σ into $\mathbb{N}$. The set of symbols of rank n is denoted by Σ_n. The elements of rank 0 are called *constants*. A *tree* t over Σ is inductively defined as follows: $t = a$, $t = f(t_1, \ldots, t_k)$ where a is any symbol in Σ_0, k is any integer satisfying $k \geq 1$, f is any symbol in Σ_k and $t_1, \ldots, t_k$ are any k trees over Σ. We denote by T_Σ the set of trees over Σ. *A tree language* is a subset of T_Σ. Let $\Sigma_{\geq 1} = \Sigma \backslash \Sigma_0$ denote the set of *non-constant symbols* of the ranked alphabet Σ. *A Finite Tree Automaton* (FTA) [5, 11] $\mathscr{A}$ is a tuple (Q, Σ, Q_T, Δ) where Q is a finite set of states, $Q_T \subset Q$ is the set of *final states* and $\Delta \subset \bigcup_{n \geq 0}(Q \times \Sigma_n \times Q^n)$ is the set of *transition rules*. This set is equivalent to the function Δ from $Q^n \times \Sigma_n$ to 2^Q defined by $(q, f, q_1, \ldots, q_n) \in \Delta \Leftrightarrow q \in \Delta(q_1, \ldots, q_n, f)$. The domain of this function can be extended to $(2^Q)^n \times \Sigma_n$ as follows: $\Delta(Q_1, \ldots, Q_n, f) = \bigcup_{(q_1, \ldots, q_n) \in Q_1 \times \cdots \times Q_n} \Delta(q_1, \ldots, q_n, f)$. Finally, we denote by Δ^* the function from $T_\Sigma \to 2^Q$ defined for any tree in T_Σ as follows: $\Delta^*(t) = \Delta(a)$ if $t = a$ with $a \in \Sigma_0$, $\Delta^*(t) = \Delta(\Delta^*(t_1), \ldots, \Delta^*(t_n), f)$ if $t = f(t_1, \ldots, t_n)$ with $f \in \Sigma_n$ and $t_1, \ldots, t_n \in T_\Sigma$. A tree is *accepted* by $\mathscr{A}$ if and only if $\Delta^*(t) \cap Q_T \neq \emptyset$. The language $\mathscr{L}(\mathscr{A})$ *recognized* by A is the set of trees accepted by $\mathscr{A}$ *i.e.* $\mathscr{L}(\mathscr{A}) = \{t \in T_\Sigma \mid \Delta^*(t) \cap Q_T \neq \emptyset\}$. Let $\sim$ be an equivalence relation over Q. We denote by $[q]$ the equivalence class of any state q in Q. The *quotient of* A w.r.t. $\sim$ is the tree automaton $A_{/\sim} = (Q_{/\sim}, \Sigma, Q_{T/\sim}, \Delta_{/\sim})$ where: $Q_{/\sim} = \{[q] \mid q \in Q\}$, $Q_{T/\sim} = \{[q] \mid q \in Q_T\}$, $\Delta_{/\sim} =$

$\{([q],f,[q_1],\ldots,[q_n]) \mid (q,f,q_1,\ldots,q_n) \in \Delta\}$. Notice that a transition $([q],f,[q_1],\ldots,[q_n])$ in $\Delta_{/\sim}$ does not imply a transition $(q,f,q_1,\ldots,q_n)$ in Δ. Moreover, the relation $\sim$ is not necessarily a congruence w.r.t. the transition function: in this paper, we will deal with specific equivalence relations (similarity relations) that turn to be congruences. This particular considerations will be clarified in Subsection 3.2.

For any integer $n \geq 0$, for any n languages $L_1,\ldots,L_n \subset T_\Sigma$, and for any symbol $f \in \Sigma_n$, $f(L_1,\ldots,L_n)$ is the tree language $\{f(t_1,\ldots,t_n) \mid t_i \in L_i\}$. The *tree substitution* of a constant c in Σ by a language $L \subset T_\Sigma$ in a tree $t \in T_\Sigma$, denoted by $t\{c \leftarrow L\}$, is the language inductively defined by: L if $t = c$; $\{d\}$ if $t = d$ where $d \in \Sigma_0 \setminus \{c\}$; $f(t_1\{c \leftarrow L\},\ldots,t_n\{c \leftarrow L\})$ if $t = f(t_1,\ldots,t_n)$ with $f \in \Sigma_n$ and $t_1,\ldots,t_n$ any n trees over Σ. Let c be a symbol in Σ_0. The *c-product* $L_1 \cdot_c L_2$ of two languages $L_1, L_2 \subset T_\Sigma$ is defined by $L_1 \cdot_c L_2 = \bigcup_{t \in L_1} \{t\{c \leftarrow L_2\}\}$. The *iterated c-product* is inductively defined for $L \subset T_\Sigma$ by: $L^{0_c} = \{c\}$ and $L^{(n+1)_c} = L^{n_c} \cup L \cdot_c L^{n_c}$. The *c-closure* of L is defined by $L^{*_c} = \bigcup_{n \geq 0} L^{n_c}$.

A *regular expression* over a ranked alphabet Σ is inductively defined by $\mathrm{E} = 0$, $\mathrm{E} \in \Sigma_0$, $\mathrm{E} = f(\mathrm{E}_1,\cdots,\mathrm{E}_n)$, $\mathrm{E} = (\mathrm{E}_1 + \mathrm{E}_2)$, $\mathrm{E} = (\mathrm{E}_1 \cdot_c \mathrm{E}_2)$, $\mathrm{E} = (\mathrm{E}_1{}^{*_c})$, where $c \in \Sigma_0$, $n \in \mathbb{N}$, $f \in \Sigma_n$ and $\mathrm{E}_1, \mathrm{E}_2, \ldots, \mathrm{E}_n$ are any n regular expressions over Σ. Parenthesis can be omitted when there is no ambiguity. We write $\mathrm{E}_1 = \mathrm{E}_2$ if E_1 and E_2 graphically coincide. We denote by $\mathrm{RegExp}(\Sigma)$ the set of all regular expressions over Σ. Every regular expression E can be seen as a tree over the ranked alphabet $\Sigma \cup \{+, \cdot_c, *_c \mid c \in \Sigma_0\}$ where $+$ and $\cdot_c$ can be seen as symbols of rank 2 and $*_c$ has rank 1. This tree is the syntax-tree T_E of E. The *alphabetical width* $||\mathrm{E}||$ of E is the number of occurrences of symbols of Σ in E. *The size* $|\mathrm{E}|$ of E is the size of its syntax tree T_E. The *language* $[\![\mathrm{E}]\!]$ *denoted by* E is inductively defined by $[\![0]\!] = \emptyset$, $[\![c]\!] = \{c\}$, $[\![f(\mathrm{E}_1,\mathrm{E}_2,\cdots,\mathrm{E}_n)]\!] = f([\![\mathrm{E}_1]\!],\ldots,[\![\mathrm{E}_n]\!])$, $[\![\mathrm{E}_1 + \mathrm{E}_2]\!] = [\![\mathrm{E}_1]\!] \cup [\![\mathrm{E}_2]\!]$, $[\![\mathrm{E}_1 \cdot_c \mathrm{E}_2]\!] = [\![\mathrm{E}_1]\!] \cdot_c [\![\mathrm{E}_2]\!]$, $[\![\mathrm{E}_1{}^{*_c}]\!] = [\![\mathrm{E}_1]\!]^{*_c}$ where $n \in \mathbb{N}$, $\mathrm{E}_1, \mathrm{E}_2, \ldots, \mathrm{E}_n$ are any n regular expressions, $f \in \Sigma_n$ and $c \in \Sigma_0$. It is well known that a tree language is accepted by some tree automaton if and only if it can be denoted by a regular expression [5, 11]. A regular expression E defined over Σ is *linear* if every symbol of rank greater than 1 appears at most once in E. Note that any constant symbol may occur more than once. Let E be a regular expression over Σ. The *linearized regular expression* $\overline{\mathrm{E}}$ in E of a regular expression E is obtained from E by marking differently all symbols of a rank greater than or equal to 1 (symbols of $\Sigma_{\geq 1}$). The marked symbols form together with the constants in Σ_0 a ranked alphabet $\mathrm{Pos}_E(E)$ the symbols of which we call *positions*. The mapping h is defined from $\mathrm{Pos}_\mathrm{E}(\mathrm{E})$ to Σ with $h(\mathrm{Pos}_\mathrm{E}(\mathrm{E})_m) \subset \Sigma_m$ for every $m \in \mathbb{N}$. It associates with a marked symbol $f_j \in \mathrm{Pos}_\mathrm{E}(\mathrm{E})_{\geq 1}$ the symbol $f \in \Sigma_{\geq 1}$ and for a symbol $c \in \Sigma_0$ the symbol $h(c) = c$. We can extend the mapping h naturally to $\mathrm{RegExp}(\mathrm{Pos}_\mathrm{E}(\mathrm{E})) \to \mathrm{RegExp}(\Sigma)$ by $h(a) = a$, $h(\mathrm{E}_1 + \mathrm{E}_2) = h(\mathrm{E}_1) + h(\mathrm{E}_2)$, $h(\mathrm{E}_1 \cdot_c \mathrm{E}_2) = h(\mathrm{E}_1) \cdot_c h(\mathrm{E}_2)$, $h(\mathrm{E}_1^{*_c}) = h(\mathrm{E}_1)^{*_c}$, $h(f_j(\mathrm{E}_1,\ldots,\mathrm{E}_n)) = f(h(\mathrm{E}_1),\ldots,h(\mathrm{E}_n))$, with $n \in \mathbb{N}$, $a \in \Sigma_0$, $f \in \Sigma_n$, $f_j \in \mathrm{Pos}_\mathrm{E}(\mathrm{E})_n$ such that $h(f_j) = f$ and $\mathrm{E}_1,\ldots,\mathrm{E}_n$ any regular expressions over $\mathrm{Pos}_\mathrm{E}(\mathrm{E})$.

3 Tree Automata from Regular Expressions

In this section, we show how to compute from a regular expression E four 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 automaton [11] and the C-continuation automaton [14].

Regular languages defined over ranked alphabet Σ are exactly the languages denoted by a regular expression on Σ. There may exist many distinct regular expressions which denote the same regular language. Two regular expressions are said to be *equivalent* if they denote the same language. To simplify handling regular expressions, we define *trivial identities* for which regular expressions denote the same language. Let $\mathrm{E}_1 \ldots \mathrm{E}_n$ be n regular expressions over a ranked alphabet Σ and c be a symbol in

Σ_0. It can be trivially shown that:
$[\![\mathrm{E}_1+0]\!] = [\![0+\mathrm{E}_1]\!] = [\![\mathrm{E}_1]\!]$, $[\![\mathrm{E}_1 \cdot_c 0]\!] = \mathrm{E}_{1c\leftarrow 0}$, $[\![[\![0 \cdot_c \mathrm{E}_1]\!] = [\![0]\!]$, $[\![0^{*_c}]\!] = [\![c]\!]$, $[\![f(\mathrm{E}_1,\ldots,0,\ldots,\mathrm{E}_n)]\!] = [\![0]\!]$, where $\mathrm{E}_{c\leftarrow 0}$ is obtained by substituting the expression 0 to any symbol c in an expression E.
Consequently, we extend the equivalence $=$ as follows:

$$\mathrm{E}_1+0 = 0+\mathrm{E}_1 = \mathrm{E}_1,\ \mathrm{E}_1 \cdot_c 0 = E_{1c\leftarrow 0},\ 0 \cdot_c \mathrm{E}_1 = 0,\ 0^{*_c} = c,\ f(\mathrm{E}_1,\ldots,0,\ldots,\mathrm{E}_n) = 0.$$

It is easy to see that these equalities preserve the language. Consequently, any regular expression E denotes the same language as a regular expression E' with no occurrence of 0 in E' or $\mathrm{E}' = 0$.

In the following of this section, E is a regular expression over a ranked alphabet Σ. The set of symbols in Σ that appear in an expression F is denoted by Σ_F.

3.1 The *K*-Position Tree Automaton

In this section, we show how to compute the K-position tree automaton of a regular expression E, recognizing $[\![E]\!]$. This is an extension of the well-known position automaton [8] for word regular expressions where the K represents the fact that any k-ary symbol is no longer a state of the automaton, but is exploded into k states. The same method was presented independently by McNaughton and Yamada [13]. Its computation is based on the computations of particular *position functions*, defined in the following.

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 $t = f(t_1,\ldots,t_n)$ be a tree. We denote by root(t) the root of t, by k-child(t) the k^{th} child of f in t, that is the root of t_k if it exists, and by Leaves(t) the set of the leaves of t, *i.e.* $\{s \in \Sigma_0 \mid s \preccurlyeq t\}$.

Let E be linear, $1 \leq k \leq m$ 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$(\mathrm{E}, f, k)$ 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 Last(E) $= \bigcup_{t\in[\![\mathrm{E}]\!]} \mathrm{Leaves}(t)$.

Example 1. *Let* $\Sigma = \Sigma_0 \cup \Sigma_1 \cup \Sigma_2$ *be defined by* $\Sigma_0 = \{a,b,c\}$, $\Sigma_1 = \{f,h\}$ *and* $\Sigma_2 = \{g\}$. *Let us consider the regular expression* E *and its linearized form defined by:*

$$\mathrm{E} = (f(a)^{*_a} \cdot_a b + h(b))^{*_b} + g(c,a)^{*_c} \cdot_c (f(a)^{*_a} \cdot_a b + h(b))^{*_b},$$
$$\overline{\mathrm{E}} = (f_1(a)^{*_a} \cdot_a b + h_2(b))^{*_b} + g_3(c,a)^{*_c} \cdot_c (f_4(a)^{*_a} \cdot_a b + h_5(b))^{*_b}.$$

The language denoted by $\overline{\mathrm{E}}$ *is* $[\![\overline{\mathrm{E}}]\!] = \{b, f_1(b), f_1(f_1(b)), f_1(h_2(b)), h_2(b), h_2(f_1(b)), h_2(h_2(b)), \ldots, g_3(b,a), g_3(g_3(b,a),a), g_3(f_4(b),a), g_3(h_5(b),a), f_4(f_4(b)), f_4(h_5(b), h_5(f_4(b)), h_5(h_5(b)), \ldots\}$.

Consequently, $\mathrm{First}(\overline{\mathrm{E}}) = \{b, f_1, h_2, g_3, f_4, h_5\}$ *and* $\mathrm{Follow}(\overline{\mathrm{E}}, f_1, 1) = \{b, f_1, h_2\}$, $\mathrm{Follow}(\overline{\mathrm{E}}, h_2, 1) = \{b, f_1, h_2\}$, $\mathrm{Follow}(\overline{\mathrm{E}}, g_3, 1) = \{b, g_3, f_4, h_5\}$, $\mathrm{Follow}(\overline{\mathrm{E}}, g_3, 2) = \{a\}$, $\mathrm{Follow}(\overline{\mathrm{E}}, f_4, 1) = \{b, f_4, h_5\}$, $\mathrm{Follow}(\overline{\mathrm{E}}, h_5, 1) = \{b, f_4, h_5\}$.

Let us first show that the position functions First and Follow are inductively computable.

Lemma 1. *Let* E *be linear. The set* First(E) *can be computed as follows:*

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

Lemma 2. *Let* E *be linear,* $1 \leq k \leq m$ *be two integers and* f *be a symbol in* Σ_m. *The set of symbols* $\mathrm{Follow}(\mathrm{E}, f, k)$ *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.

Definition 1. *Let* E *be linear. The* K-position automaton $\mathscr{P}_{\mathrm{E}}$ *is the automaton* (Q,Σ,Q_T,Δ) *defined by*

$$\begin{aligned}Q=&\{f^k\mid f\in\Sigma_m\wedge 1\le k\le m\}\cup\{\varepsilon^1\}\text{ with }\varepsilon^1\text{ a new symbol not in }\Sigma,\ Q_T=\{\varepsilon^1\},\\ \Delta=&\quad\{(f^k,g,g^1,\ldots,g^n)\mid f\in\Sigma_m\wedge k\le m\wedge g\in\Sigma_n\wedge g\in\mathrm{Follow}(\mathrm{E},f,k)\}\\ &\cup\{(\varepsilon^1,f,f^1,\ldots,f^m)\mid f\in\Sigma_m\wedge f\in\mathrm{First}(\mathrm{E})\}\\ &\cup\{(\varepsilon^1,c)\mid c\in\Sigma_0\wedge c\in\mathrm{First}(\mathrm{E})\}\\ &\cup\{(f^k,c)\mid f\in\Sigma_m\wedge k\le m\wedge c\in\mathrm{Follow}(\mathrm{E},f,k)\}\end{aligned}$$

In order to show that the K-position tree automaton of E accepts $[\![\mathrm{E}]\!]$, we characterize the membership of a tree t in the language denoted by E using the functions First and Follow.

Proposition 1. *Let* E *be linear. A tree* t *belongs to* $[\![\mathrm{E}]\!]$ *if and only if:*

1. $\mathrm{root}(t)\in\mathrm{First}(\mathrm{E})$ *and*
2. *for every subtree* $f(t_1,\ldots,t_m)$ *of* t*, for any integer* k *in* $\{1,\ldots,m\}$, $\mathrm{root}(t_k)\in\mathrm{Follow}(\mathrm{E},f,k)$.

Let us show how to link the characterization in Proposition 1 with the transition sequences in $\mathscr{P}_{\mathrm{E}}$.

Proposition 2. *Let* E *be linear and* $\mathscr{P}_{\mathrm{E}}=(Q,\Sigma,Q_T,\Delta)$. *Let* $t=f(t_1,\ldots,t_m)$ *be a term in* T_Σ. *Then the two following propositions are equivalent:*

1. $\forall g(s_1,\ldots,s_l)\preccurlyeq t$, $\forall p\le l$, $\mathrm{root}(s_p)\in\mathrm{Follow}(\mathrm{E},g,p)$,
2. $\forall 1\le k\le m$, $f^k\in\Delta^*(t_k)$.

As a direct consequence of the two previous propositions, it can be shown that the K-position automaton of E recognizes the language denoted by E.

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

This construction can be extended to expressions that are not necessarily linear using the linearization and the mapping h. The *K-Position Automaton* $\mathscr{P}_{\mathrm{E}}$ associated with E is obtained by replacing each transition $(f_j^k,g_i,g_i^1,\ldots,g_i^n)$ of the tree automaton $\mathscr{P}_{\overline{\mathrm{E}}}$ by $(f_j^k,h(g_i),g_i^1,\ldots,g_i^n)$.

Corollary 1. $h([\![\overline{\mathrm{E}}]\!])=h(\mathscr{L}(\mathscr{P}_{\overline{\mathrm{E}}}))=\mathscr{L}(\mathscr{P}_{\mathrm{E}})=[\![\mathrm{E}]\!]$.

Example 2. *Let* $\mathrm{E}=(f(a)^{*_a}\cdot_a b+h(b))^{*_b}+g(c,a)^{*_c}\cdot_c(f(a)^{*_a}\cdot_a b+h(b))^{*_b}$ *be the regular expression of Example 1. The k-Position Automaton* $\mathscr{P}_{\overline{\mathrm{E}}}$ *associated with* $\overline{\mathrm{E}}$ *is given in Figure 1. The set of states is* $Q=\{\varepsilon^1,f_1^1,h_2^1,g_3^1,g_3^2,f_4^1,h_5^1\}$. *The set of final states is* $Q_T=\{\varepsilon^1\}$. *The set of transition rules* Δ *is*

$$\begin{array}{lllllll}f_1(f_1^1)\to f_1^1 & f_1(f_1^1)\to\varepsilon^1 & f_1(h_2^1)\to\varepsilon^1 & f_1(h_2^1)\to f_1^1 & h_2(f_1^1)\to\varepsilon^1 & b\to f_1^1 & b\to h_2^1\\ h_2(f_1^1)\to h_2^1 & h_2(h_2^1)\to\varepsilon^1 & h_2(h_2^1)\to h_2^1 & g_3(f_4^1,g_3^2)\to\varepsilon^1 & g_3(h_5^1,g_3^2)\to\varepsilon^1 & b\to g_3^1 & a\to g_3^2\\ f_4(f_4^1)\to\varepsilon^1 & f_4(f_4^1)\to f_4^1 & f_4(h_5^1)\to\varepsilon^1 & f_4(h_5^1)\to f_4^1 & h_5(f_4^1)\to\varepsilon^1 & b\to h_5^1 & b\to f_4^1\\ b\to\varepsilon^1 & h_5(f_4^1)\to h_5^1 & h_5(h_5^1)\to h_5^1 & g_3(g_3^1,g_3^2)\to\varepsilon^1 & h_5(h_5^1)\to\varepsilon^1 & & \end{array}$$

The number of states is $|Q|=7$ *and the number of transition rules is* $|\Delta|=26$.

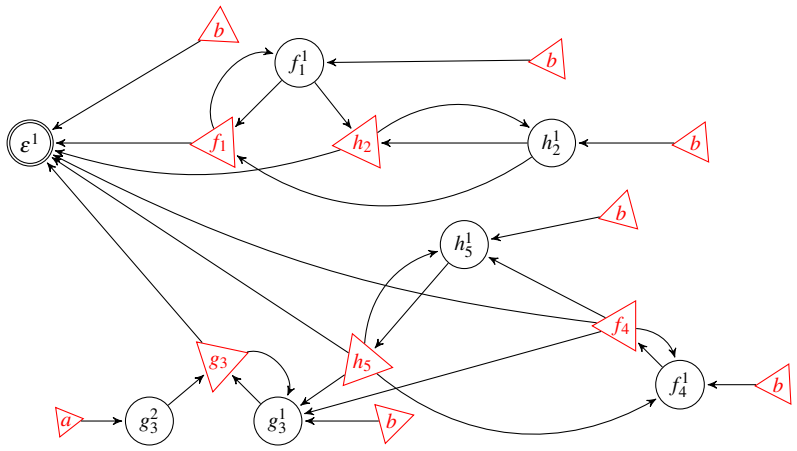

Figure 1: The k-Position Automaton $\mathscr{P}_{\mathrm{E}}$.

3.2 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 [9] in the case of words, and that it is a quotient of the K-position automaton, similarly to the case of words. Notice that in this automaton, states are no longer positions, but sets of positions.

Definition 2. *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*

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

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

The quotient from $\mathscr{P}_E$ to $\mathscr{F}_E$ is defined by the following similarity relation. Notice that we extend the definition of the function Follow to the position ε^1 by $\mathrm{Follow}(E, \varepsilon^1, 1) = \mathrm{First}(E)$. Let E be linear and $\mathscr{P}_{\mathrm{E}} = (Q, \Sigma, Q_T, \Delta)$. The *Follow Relation* is the relation $\sim_{\mathscr{F}}$ defined for any two states f^k and g^l in Q by $f^k \sim_{\mathscr{F}} g^l \Leftrightarrow \mathrm{Follow}(\mathrm{E}, f, k) = \mathrm{Follow}(\mathrm{E}, g, l)$.

Proposition 4. *Let E be linear. The relation* $\sim_{\mathscr{F}}$ *is the largest similarity relation over* $\mathscr{P}_E$.

Proposition 5. *Let* E *be linear. The finite tree automaton* $\mathscr{P}_{\mathrm{E}}/_{\sim_{\mathscr{F}}}$ *is isomorphic to* $\mathscr{F}_{\mathrm{E}}$.

As a direct consequence of the previous results, the following theorem can be shown.

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

Finally, this method can be extended to expressions that are not necessarily linear as follows. The *Follow Automaton* $\mathscr{F}_{\mathrm{E}}$ associated with E is obtained by replacing each transition $(I, f_j, \mathrm{Follow}(\mathrm{E}, f_j, 1), \ldots, \mathrm{Follow}(\mathrm{E}, f_j, m))$ of $\mathscr{F}_{\overline{\mathrm{E}}}$ by $(I, h(f_j), \mathrm{Follow}(\mathrm{E}, f_j, 1), \ldots, \mathrm{Follow}(\mathrm{E}, f_j, m))$.

Corollary 2. $\mathscr{L}(\mathscr{F}_{\mathrm{E}}) = [\![\mathrm{E}]\!]$.

Example 3. *The Follow Automaton* $\mathscr{F}_{\mathrm{E}}$ *associated with* $\mathrm{E} = (f(a)^{*_a} \cdot_a b + h(b))^{*_b} + g(c,a)^{*_c} \cdot_c (f(a)^{*_a} \cdot_a b + h(b))^{*_b}$ *of Example 1 is given in Figure 2.*

The set of states is $Q = \{\{a\}, \{b, f_1, h_2\}, \{b, f_1, h_2, g_3, f_4, h_5\}, \{b, g_3, f_4, h_5\}, \{b, f_4, h_5\}\}$ *and* $Q_T = \{\{b, f_1, h_2, g_3, f_4, h_5\}\}$. *The set of transition rules* Δ *is*

$f(\{b, f_1, h_2\}) \to \{b, f_1, h_2\}$ $\quad h(\{b, f_1, h_2\}) \to \{b, f_1, h_2, g_3, f_4, h_5\}$ $\quad b \to \{b, f_1, h_2, g_3, f_4, h_5\}$
$h(\{b, f_1, h_2\}) \to \{b, f_1, h_2\}$ $\quad f(\{b, f_4, h_5\}) \to \{b, f_4, h_5\}$ $\quad b \to \{b, g_3, f_4, h_5\}$
$f(\{b, f_4, h_5\}) \to \{b, g_3, f_4, h_5\}$ $\quad f(\{b, f_4, h_5\}) \to \{b, f_1, h_2, g_3, f_4, h_5\}$ $\quad a \to \{a\}$
$h(\{b, f_4, h_5\}) \to \{b, f_1, h_2, g_3, f_4, h_5\}$ $\quad h(\{b, f_4, h_5\}) \to \{b, f_4, h_5\}$ $\quad b \to \{b, f_1, h_2\}$
$h(\{b, f_4, h_5\}) \to \{b, g_3, f_4, h_5\}$ $\quad f(\{b, f_1, h_2\}) \to \{b, f_1, h_2, g_3, f_4, h_5\}$ $\quad b \to \{b, f_4, h_5\}$
$g(\{b, g_3, f_4, h_5\}, \{a\}) \to \{b, g_3, f_4, h_5\}$ $\quad g(\{b, g_3, f_4, h_5\}, \{a\}) \to \{b, f_1, h_2, g_3, f_4, h_5\}$

The number of states is $|Q| = 5$ *and the number of transition rules is* $|\Delta| = 17$.

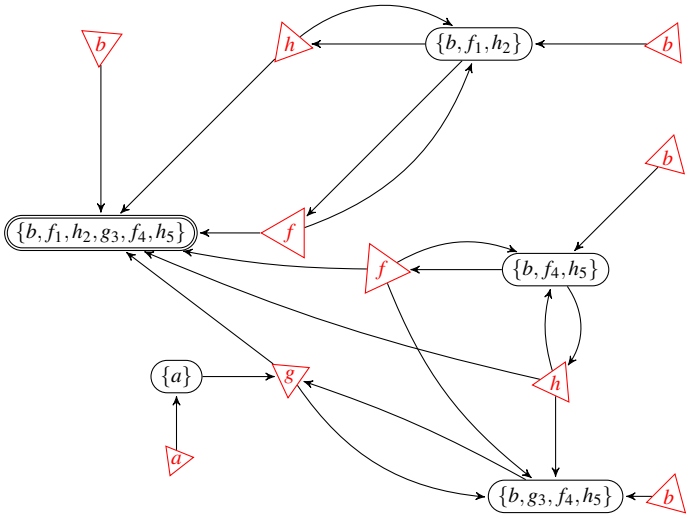

Figure 2: The Follow Automaton $\mathscr{F}_{\mathrm{E}}$.

3.3 The Equation Tree Automaton

In [11], 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:

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

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

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

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

Example 4. *Let* $\mathrm{E} = \underbrace{(f(a)^{*_a} \cdot_a b + h(b))^{*_b}}_{\mathrm{F}} + \underbrace{g(c,a)^{*_c}}_{\mathrm{G}} \cdot_c \underbrace{(f(a)^{*_a} \cdot_a b + h(b))^{*_b}}_{\mathrm{F}}$ *(Example 1).*

$$\begin{array}{lll} \partial_h(\mathrm{E}) = \{b \cdot_b \mathrm{F}\}, & \partial_f(\mathrm{E}) = \{((a \cdot_a f(a)^{*_a}) \cdot_a b) \cdot_b \mathrm{F}\} & \partial_{ff}(\mathrm{E}) = \{((a \cdot_a f(a)^{*_a}) \cdot_a b) \cdot_b \mathrm{F}\}, \\ \partial_{fh}(\mathrm{E}) = \{b \cdot_b \mathrm{F}\}, & \partial_g(\mathrm{E}) = \{(a \cdot_c \mathrm{G}) \cdot_c \mathrm{F},\ (c \cdot_c \mathrm{G}) \cdot_c \mathrm{F}\} & \partial_{hf}(\mathrm{E}) = \{((a \cdot_a f(a)^{*_a}) \cdot_a b) \cdot_b \mathrm{F}\} \\ \partial_{gh}(\mathrm{E}) = \{b \cdot_b \mathrm{F}\} & \partial_{hh}(\mathrm{E}) = \{((a \cdot_a f(a)^{*_a}) \cdot_a b) \cdot_b \mathrm{F}\}, & \partial_{gf}(\mathrm{E}) = \{((a \cdot_a f(a)^{*_a}) \cdot_a b) \cdot_b \mathrm{F}\} \\ & \partial_{gg}(\mathrm{E}) = \{(a \cdot_c \mathrm{G}) \cdot_c \mathrm{F},\ (c \cdot_c \mathrm{G}) \cdot_c \mathrm{F}\}, & \end{array}$$

The set of states Q *is* $q_0 = \mathrm{E}$, $q_1 = ((a \cdot_a f(a)^{*_a}) \cdot_a b) \cdot_b \mathrm{F}$, $q_2 = b \cdot_b \mathrm{F}$, $q_3 = (c \cdot_c \mathrm{G}) \cdot_c \mathrm{F}$, $q_4 = (a \cdot_c \mathrm{G}) \cdot_c \mathrm{F}$. *The set of final states is* $Q_T = \{q_0\}$. *The set of transition rules is*

$$\begin{array}{lllll} b \to q_0 & b \to q_1 & b \to q_3 & b \to q_2 & f(q_1) \to q_0 \\ h(q_2) \to q_0 & g(q_3, q_4) \to q_0 & h(q_2) \to q_1 & g(q_3, q_4) \to q_4 & f(q_1) \to q_1 \\ h(q_2) \to q_2 & f(q_1) \to q_2 & f(q_1) \to q_4 & h(q_2) \to q_4 & a \to q_4 \end{array}$$

The number of states is $|Q| = 5$ *and the number of transition rules is* $|\Delta| = 15$. *The Equation Automaton associated with* E *is given in Figure 3.*

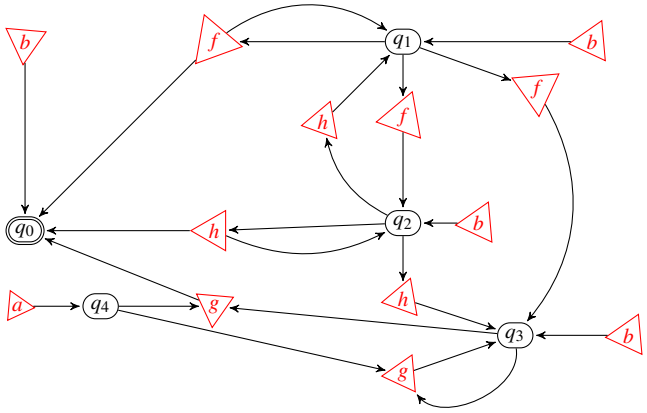

Figure 3: The Equation Automaton $\mathscr{A}_{\mathrm{E}}$.

3.4 The k-C-Continuation Tree Automaton

In [11], 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, 10]. In [14, 15], we show how to inductively compute them. We also show how to efficiently compute the k-C-Continuation tree automaton associated with a regular expression. In this section, we prove that this automaton is isomorphic to the k-position tree automaton, similarly to the case of words.

Definition 3 ([14, 15]). *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:*

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

$$C_{f^k}(\mathrm{E}_1 + \mathrm{E}_2) = \begin{cases} C_{f^k}(E_1) & \text{if } f \in \Sigma_{E_1} \\ C_{f^k}(E_2) & \text{if } f \in \Sigma_{E_2} \end{cases}$$

$$C_{f^k}(E_1 \cdot_c E_2) = \begin{cases} C_{f^k}(E_1) \cdot_c \mathrm{G} & \text{if } f \in \Sigma_{E_1} \\ C_{f^k}(E_2) & \text{if } f \in \Sigma_{E_2} \\ & \text{and } c \in \mathrm{Last}(E_1) \\ 0 & \text{otherwise} \end{cases}$$

$$C_{f^k}(\mathrm{F}^{*_c}) = C_{f^k}(\mathrm{F}) \cdot_c \mathrm{F}^{*_c}$$

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

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

Definition 4 ([14, 15]). *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 by*

- $Q_{\mathscr{C}} = \{(f^k, C_{f^k}(\mathrm{E})) \mid f \in \Sigma_m, 1 \leq k \leq m\} \cup \{(\varepsilon^1, C_{\varepsilon^1}(\mathrm{E}))\}$,
- $\Delta_{\mathscr{C}} = \{((x, C_x(\mathrm{E})), g, ((g^1, C_{g^1}(\mathrm{E})), \ldots, (g^m, C_{g^m}(\mathrm{E})))) \mid g \in \Sigma_{Em},$
  $m \geq 1, (C_{g^1}(\mathrm{E}), \ldots, C_{g^m}(\mathrm{E})) \in g^{-1}(C_x(\mathrm{E}))\}$
  $\cup \{((x, C_x(\mathrm{E})), c) \mid, c \in [\![C_x(\mathrm{E})]\!] \cap \Sigma_0\}$

The *C-Continuation tree automaton* $\mathscr{C}_{\mathrm{E}}$ associated with E is obtained by relabelling the transitions of $\mathscr{C}_{\overline{\mathrm{E}}}$ using the mapping h.

Theorem 3 ([14, 15]). *The automaton* $\mathscr{C}_{\mathrm{E}}$ *accepts* $[\![\mathrm{E}]\!]$.

Example 5. *Let* $\mathrm{E} = (f(a)^{*_a} \cdot_a b + h(b))^{*_b} + g(c,a)^{*_c} \cdot_c (f(a)^{*_a} \cdot_a b + h(b))^{*_b}$ *defined in Example 1 and* $\overline{\mathrm{E}} = \underbrace{(f_1(a)^{*_a} \cdot_a b + h_2(b))^{*_b}}_{\mathrm{F}_1} + \underbrace{g_3(c,a)^{*_c}}_{\mathrm{G}_2} \cdot_c \underbrace{(f_4(a)^{*_a} \cdot_a b + h_5(b))^{*_b}}_{\mathrm{F}_3}$.

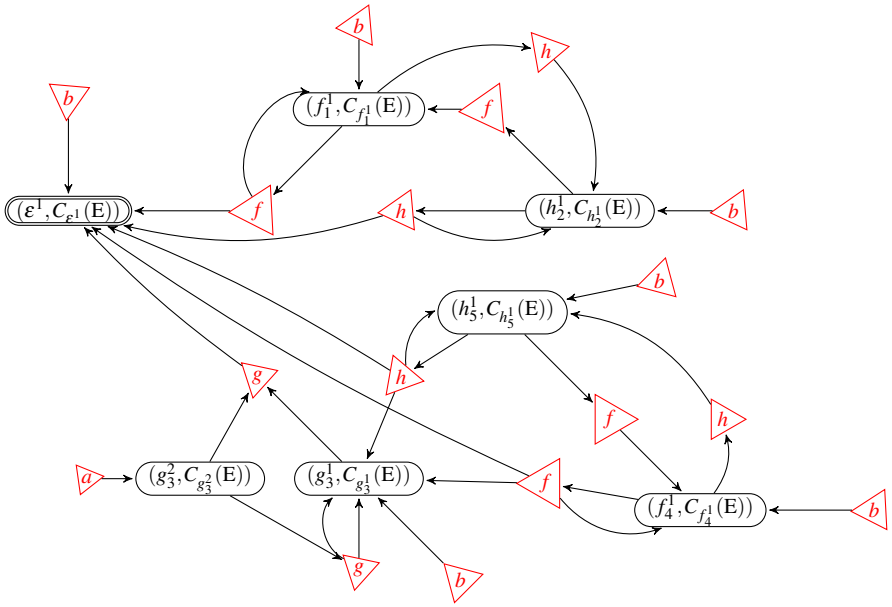

Figure 4: The k-C-Continuation Automaton $\mathscr{C}_{\mathrm{E}}$.

The computation of the k-C-Continuations of E *using the Definition 3 is given in Table 1.*

$$
\begin{array}{ll}
C_{f_1^1}(\overline{\mathrm{E}}) = ((a \cdot_a f_1(a)^{*_a}) \cdot_a b) \cdot_b \mathrm{F}_1 & h(C_{f_1^1}(\overline{\mathrm{E}})) = ((a \cdot_a f(a)^{*_a}) \cdot_a b) \cdot_b \mathrm{F}, \\
C_{h_2^1}(\overline{\mathrm{E}}) = b \cdot_b \mathrm{F}_1 & h(C_{h_2^1}(\overline{\mathrm{E}})) = b \cdot_b \mathrm{F}, \\
C_{g_3^1}(\overline{\mathrm{E}}) = (c \cdot_c g_3(c,a)^{*_c}) \cdot_c \mathrm{F}_3 & h(C_{g_3^1}(\overline{\mathrm{E}})) = (c \cdot_c g(c,a)^{*_c}) \cdot_c \mathrm{F}, \\
C_{g_3^2}(\overline{\mathrm{E}}) = (a \cdot_c g_3(c,a))^{*_c}) \cdot_c \mathrm{F}_3 & h(C_{g_3^2}(\overline{\mathrm{E}})) = (a \cdot_c g(c,a))^{*_c}) \cdot_c \mathrm{F}, \\
C_{f_4^1}(\overline{\mathrm{E}}) = ((a \cdot_a f_4(a)^{*_a}) \cdot_a b) \cdot_b \mathrm{F}_3 & h(C_{f_4^1}(\overline{\mathrm{E}})) = ((a \cdot_a f(a)^{*_a}) \cdot_a b) \cdot_b \mathrm{F}, \\
C_{h_5^1}(\overline{\mathrm{E}}) = b \cdot_b \mathrm{F}_3 & h(C_{h_5^1}(\overline{\mathrm{E}})) = b \cdot_b \mathrm{F}.
\end{array}
$$

Table 1: The k-C-Continuations of $\overline{\mathrm{E}}$.

The set of states of the automaton $\mathscr{C}_{\mathrm{E}}$ *is* $Q = \{(\varepsilon^1, C_{\varepsilon^1}(\overline{\mathrm{E}})), (f_1^1, C_{f_1^1}(\overline{\mathrm{E}})), (h_2^1, C_{h_2^1}(\overline{\mathrm{E}})),$
$(g_3^1, C_{g_3^1}(\overline{\mathrm{E}})), (g_3^2, C_{g_3^2}(\overline{\mathrm{E}})), (f_4^1, C_{f_4^1}(\overline{\mathrm{E}})), (h_5^1, C_{h_5^1}(\overline{\mathrm{E}}))\}$.

The set of transition rules Δ *is*

$$
\begin{array}{lll}
f((f_1^1, C_{f_1^1}(\overline{\mathrm{E}}))) \to (\varepsilon^1, C_{\varepsilon^1}(\mathrm{E})) & f((f_4^1, C_{f_4^1}(\overline{\mathrm{E}}))) \to (\varepsilon^1, C_{\varepsilon^1}(\mathrm{E})) & b \to (f_1^1, C_{f_1^1}(\overline{\mathrm{E}})) \\
g((g_3^1, C_{g_3^1}(\overline{\mathrm{E}})), (g_3^2, C_{g_3^2}(\overline{\mathrm{E}}))) \to (\varepsilon^1, C_{\varepsilon^1}(\mathrm{E})) & h((h_5^1, C_{h_5^1}(\overline{\mathrm{E}}))) \to (\varepsilon^1, C_{\varepsilon^1}(\mathrm{E})) & b \to (\varepsilon^1, C_{\varepsilon^1}(\mathrm{E})) \\
f((f_1^1, C_{f_1^1}(\overline{\mathrm{E}}))) \to (f_1^1, C_{f_1^1}(\overline{\mathrm{E}})) & h((h_2^1, C_{h_2^1}(\overline{\mathrm{E}}))) \to (h_2^1, C_{h_2^1}(\overline{\mathrm{E}})) & b \to (g_3^1, C_{g_3^1}(\overline{\mathrm{E}})) \\
h((h_2^1, C_{h_2^1}(\overline{\mathrm{E}}))) \to (\varepsilon^1, C_{\varepsilon^1}(\mathrm{E})) & f((f_1^1, C_{f_1^1}(\overline{\mathrm{E}}))) \to (h_2^1, C_{h_2^1}(\overline{\mathrm{E}})) & b \to (h_2^1, C_{h_2^1}(\overline{\mathrm{E}})) \\
f((f_4^1, C_{f_4^1}(\overline{\mathrm{E}}))) \to (f_4^1, C_{f_4^1}(\overline{\mathrm{E}})) & f((f_4^1, C_{f_4^1}(\overline{\mathrm{E}}))) \to (h_5^1, C_{h_5^1}(\overline{\mathrm{E}})) & a \to (g_3^2, C_{g_3^2}(\overline{\mathrm{E}})) \\
h((h_2^1, C_{h_2^1}(\overline{\mathrm{E}}))) \to (f_1^1, C_{f_1^1}(\overline{\mathrm{E}})) & h((h_5^1, C_{h_5^1}(\overline{\mathrm{E}}))) \to (h_5^1, C_{h_5^1}(\overline{\mathrm{E}})) & b \to (h_5^1, C_{h_5^1}(\overline{\mathrm{E}})) \\
f((f_4^1, C_{f_4^1}(\overline{\mathrm{E}}))) \to (g_3^1, C_{g_3^1}(\overline{\mathrm{E}})) & h((h_5^1, C_{h_5^1}(\overline{\mathrm{E}}))) \to (f_4^1, C_{f_4^1}(\overline{\mathrm{E}})) & b \to (f_4^1, C_{f_4^1}(\overline{\mathrm{E}})) \\
g((g_3^1, C_{g_3^1}(\overline{\mathrm{E}})), (g_3^2, C_{g_3^2}(\overline{\mathrm{E}}))) \to (g_3^1, C_{g_3^1}(\overline{\mathrm{E}})) & h((h_5^1, C_{h_5^1}(\overline{\mathrm{E}}))) \to (g_3^1, C_{g_3^1}(\overline{\mathrm{E}})) &
\end{array}
$$

The number of states is $|Q| = 5$ *and the number of transition rules is* $|\Delta| = 15$. *The k-C-Continuation Automaton associated with* E *is given in Figure 4.*

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 6 ([14, 15]). *The automaton* $\mathscr{C}_{\mathrm{E}}/_{\sim_e}$ *is isomorphic to* $\mathscr{A}_{\mathrm{E}}$.

Example 6. *Using the equivalence-relation* $\sim_e$ *over the set of states of k-C-Continuation Automaton* $\mathscr{C}_{\mathrm{E}}$ *(Figure 4) we see that* $h(C_{f_1^1}(\overline{\mathrm{E}})) = h(C_{f_4^1}(\overline{\mathrm{E}}))$ *and* $h(C_{h_2^1}(\overline{\mathrm{E}})) = h(C_{h_5^1}(\overline{\mathrm{E}}))$. *The automaton* $\mathscr{C}_{\mathrm{E}}/_{\sim_e}$ *is given in Figure 5.*

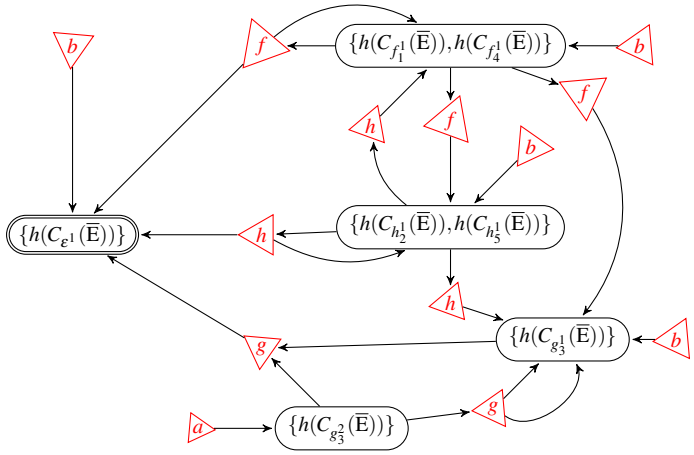

Figure 5: The Automaton $\mathscr{C}_{\mathrm{E}}/_{\sim_e}$.

In order to show that the k-C-continuation tree automaton of E is isomorphic to the k-position automaton of E, we first show the link between the position functions and the C-continuations.

Proposition 7 ([14, 15]). *Let* E *be linear,* $1 \leq k \leq m$ *be two integers and* f *be a position in* $\Sigma_E \cap \Sigma_m$. *Then* $\mathrm{Follow}(E, f, k) = \mathrm{First}(C_{f^k}(\overline{\mathrm{E}}))$.

Lemma 3. *Let* E *be linear and* g *be a symbol in* $\Sigma_{\geq 1}$. *Then* $g^{-1}(\mathrm{E}) \neq \emptyset \Leftrightarrow g \in \mathrm{First}(\mathrm{E})$.

Corollary 3. *Let* E *be linear,* $1 \leq k \leq m$ *be two integers and* f *and* g *be two symbols in* Σ. *Then,* $g^{-1}(C_{f^k}(\mathrm{E})) \neq \emptyset \Leftrightarrow g \in \mathrm{First}(C_{f^k}(\mathrm{E}))$.

Lemma 4. *Let* E *be linear,* $1 \leq k \leq m$ *be two integers and* f *and* g *be two symbols in* Σ. *Then,* $g^{-1}(C_{f^k}(\mathrm{E})) \neq \emptyset \Leftrightarrow g \in \mathrm{Follow}(\mathrm{E}, f, k)$.

Proposition 8. *Let* E *be linear. The automaton* $\mathscr{C}_{\mathrm{E}}$ *is isomorphic to* $\mathscr{P}_{\mathrm{E}}$.

This proposition can be extended to expressions that are not necessarily linear since $\mathscr{C}_{\mathrm{E}}$ and $\mathscr{P}_{\mathrm{E}}$ are relabelings of $\mathscr{C}_{\overline{\mathrm{E}}}$ and $\mathscr{P}_{\overline{\mathrm{E}}}$.

Corollary 4. *The automaton* $\mathscr{C}_{\mathrm{E}}$ *is isomorphic to* $\mathscr{P}_{\mathrm{E}}$.

We define the similarity relation denoted by $\equiv$ over the set of states of the automaton $\mathscr{C}_{\mathrm{E}}$ as follows:

$$(f^k, C_{f^k}(\overline{\mathrm{E}})) \equiv (g^p, C_{g^p}(\overline{\mathrm{E}})) \Leftrightarrow \mathrm{Follow}(\overline{\mathrm{E}}, f, k) = \mathrm{Follow}(\overline{\mathrm{E}}, g, p).$$

Corollary 5. *The finite tree automaton* $\mathscr{C}_{\mathrm{E}}/_{\equiv}$ *is isomorphic to the follow automaton* $\mathscr{F}_{\mathrm{E}}$.

4 Comparison between the Equation and the Follow Automata

We discuss in this section two examples to compare the equation and the follow automata.

Let $\Sigma = \Sigma_0 \cup \Sigma_1$ be the ranked alphabet defined by $\Sigma_0 = \{a\}$ and $\Sigma_1 = \{f_1, \ldots, f_n\}$. Let us consider the linear regular expression $\mathrm{E} = ((f_1(a)^{*_a} \cdot_a f_2(a)^{*_a}) \cdot_a \ldots) \cdot_a f_n(a)^{*_a})^{*_a}$ defined over Σ. Then the size of E is $|\mathrm{E}| = 4n-1$ and its alphabet width is $||\mathrm{E}|| = n+1$. We have $\mathrm{First}(\mathrm{E}) = \{a, f_1, f_2, \ldots, f_n\}$ and $\mathrm{Follow}(\mathrm{E}, f_1, 1) = \mathrm{Follow}(\mathrm{E}, f_2, 1) = \ldots = \mathrm{Follow}(\mathrm{E}, f_n, 1) = \{a, f_1, f_2, \ldots, f_n\}$.
The partial derivatives associated with E are:

$$\partial_{f_1}(\mathrm{E}) = \{(((a \cdot_a f_1(a)^{*_a} \cdot_a f_2(a)^{*_a}) \cdot_a \ldots) \cdot_a f_n(a)^{*_a}) \cdot_a \mathrm{E}\}$$
$$\partial_{f_2}(\mathrm{E}) = \{((a \cdot_a f_2(a)^{*_a} \cdot_a \ldots) \cdot_a f_n(a)^{*_a}) \cdot_a \mathrm{E}\}, \ldots$$

$$\partial_{f_n}(\mathrm{E}) = \{(a \cdot_a f_n(a)^{*_a}) \cdot_a \mathrm{E}\}.$$

The K-position automaton associated with E has $n+1$ states.
The follow automaton associated with E has 1 state.
The equation automaton associated with E has: $n+1$ states.

Let $\mathrm{F} = \underbrace{(f(a)^{*_a} + f(a)^{*_a} + \cdots + f(a)^{*_a})}_{f(a)^{*_a}\ n\text{-times}}$ be a regular expression defined over the ranked alphabet $\Sigma = \Sigma_0 \cup \Sigma_1$ such that $\Sigma_0 = \{a\}$ and $\Sigma_1 = \{f\}$. We have $|\mathrm{E}| = 4n-1$ and $||\mathrm{E}|| = n+1$. The linearized form associated with F is $\overline{\mathrm{F}} = (f_1(a)^{*_a} + f_2(a)^{*_a} + \cdots + f_n(a)^{*_a})$. The set First(F) $= \{a, f_1, f_2, \ldots, f_n\}$, Follow$(\overline{\mathrm{F}}, f_1, 1) = \{a, f_1\}$, Follow$(\overline{\mathrm{F}}, f_2, 1) = \{a, f_2\}, \ldots$, and Follow$(\overline{\mathrm{F}}, f_n, 1) = \{a, f_n\}$.
The partial derivatives associated with F are $\partial_f(\mathrm{F}) = \{a \cdot_a f(a)^{*_a}\}$, $\partial_{ff}(\mathrm{F}) = \{a \cdot_a f(a)^{*_a}\}$.
The K-position automaton associated with F has $n+1$ states.
The follow automaton associated with F has: $n+1$ states.
The equation automaton associated with F has: 2 states.

From these examples we state that the two automata are incomparable:

Proposition 9. *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.*

4.1 A smaller automaton

In [7] 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 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 equivalence 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.

Example 7. *Let* $\mathrm{E} = (f(a)^{*_a} \cdot_a b + h(b))^{*_b} + g(c,a)^{*_c} \cdot_c (f(a)^{*_a} \cdot_a b + h(b))^{*_b}$ *defined in Example 1 and* $\overline{\mathrm{E}} = \underbrace{(f_1(a)^{*_a} \cdot_a b + h_2(b))^{*_b}}_{\mathrm{F_1}} + \underbrace{g_3(c,a)^{*_c}}_{\mathrm{G_2}} \cdot_c \underbrace{(f_4(a)^{*_a} \cdot_a b + h_5(b))^{*_b}}_{\mathrm{F_3}}$.

$$\begin{aligned}
C_{f_1^1}(\overline{\mathrm{E}}) &= ((a \cdot_a f_1(a)^{*_a}) \cdot_a b) \cdot_b (f_1(a)^{*_a} \cdot_a b + h_2(b))^{*_b}, \\
C_{h_2^1}(\overline{\mathrm{E}}) &= b \cdot_b (f_1(a)^{*_a} \cdot_a b + h_2(b))^{*_b}, \\
C_{g_3^1}(\overline{\mathrm{E}}) &= (c \cdot_c g_3(c,a)^{*_c}) \cdot_c (f_4(a)^{*_a} \cdot_a b + h_5(b))^{*_b}, \\
C_{g_3^2}(\overline{\mathrm{E}}) &= (a \cdot_c g_3(c,a))^{*_c}) \cdot_c (f_4(a)^{*_a} \cdot_a b + h_5(b))^{*_b}, \\
C_{f_4^1}(\overline{\mathrm{E}}) &= ((a \cdot_a f_4(a)^{*_a}) \cdot_a b) \cdot_b (f_4(a)^{*_a} \cdot_a b + h_5(b))^{*_b}, \\
C_{h_5^1}(\overline{\mathrm{E}}) &= b \cdot_b (f_4(a)^{*_a} \cdot_a b + h_5(b))^{*_b}.
\end{aligned}$$

Applying $\sim_{\mathscr{F}}$ *over the states of* $\mathscr{C}_{\mathrm{E}}$ *we obtain:* $C_{f_1^1}(\overline{\mathrm{E}}) \sim_{\mathscr{F}} C_{h_2^1}(\overline{\mathrm{E}})$ *then the two states are merged,* $C_{f_4^1}(\overline{\mathrm{E}}) \sim_{\mathscr{F}} C_{h_5^1}(\overline{\mathrm{E}})$ *so they are merged. The states* $C_{g_3^1}(\overline{\mathrm{E}})$ *and* $C_{g_3^2}(\overline{\mathrm{E}})$ *are not merged with anyone.*

The number of states is $|Q| = 5$ *and the number of transition rules is* $|\Delta| = 15$.

The quotient automaton of this automaton by the equivalence relation $\sim_{\mathscr{F}}$ is given in Figure 6.

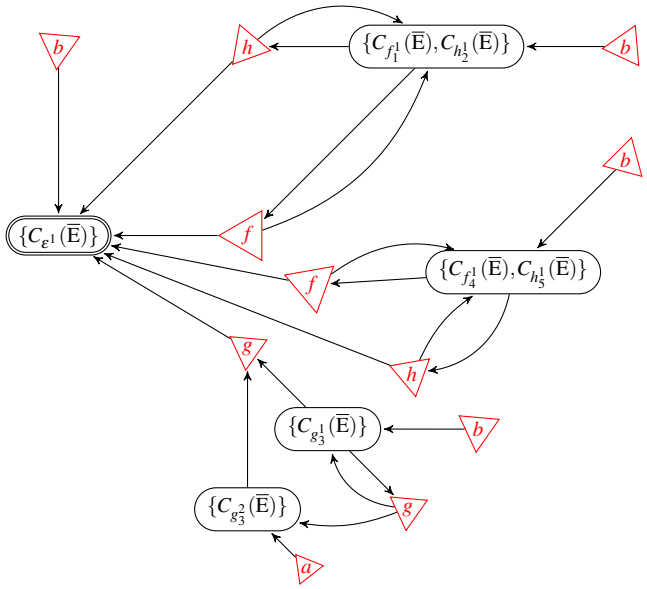

Figure 6: The Automaton $\mathscr{C}_{\mathrm{E}}/_{\sim_{\mathscr{F}}}$.

The quotient automaton of the automaton $\mathscr{C}_{\mathrm{E}}/_{\sim_{\mathscr{F}}}$ by the equivalent relation $\sim_e$ is given in Figure 7. The number of states is $|Q| = 4$ and the number of transition rules is $|\Delta| = 14$.

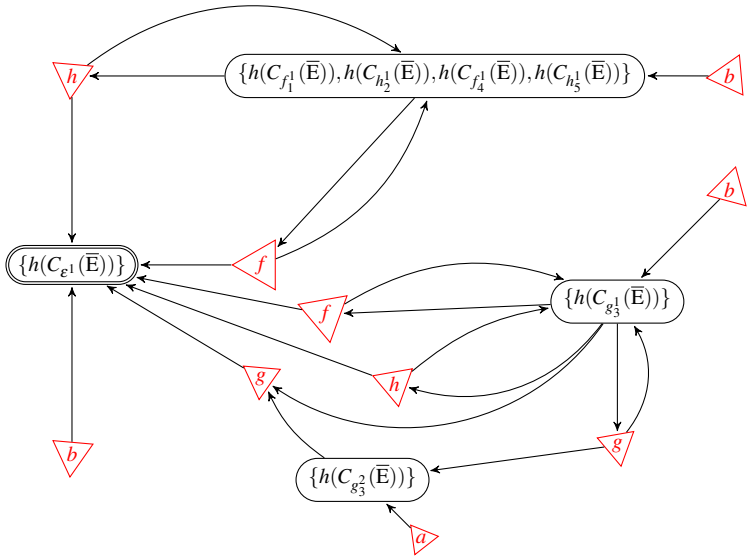

Figure 7: The resulting automaton.

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 8.

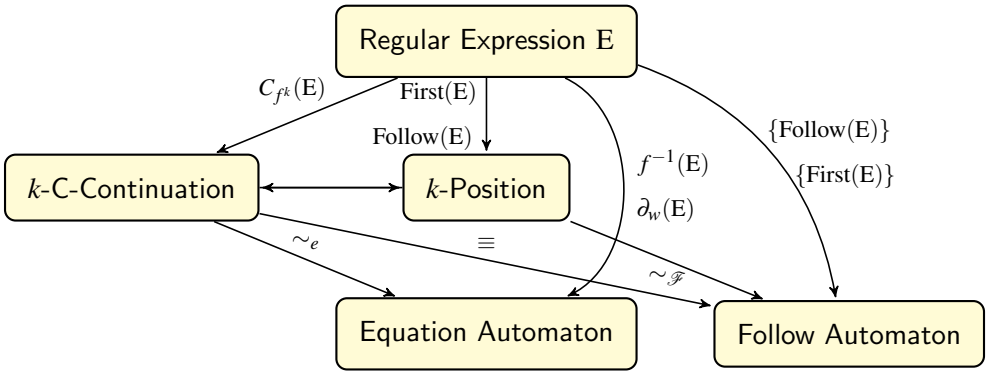

Figure 8: Relation between Automata

Looking for reductions of the set of states, we applied the algorithm by García *et al.* [7] 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.

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] H. Comon, M. Dauchet, R. Gilleron, F. Jacquemard, D. Lugiez, C. Loding, S. Tison & M. Tommasi (2007): *Tree Automata Techniques and Applications*. Available on: `http://www.grappa.univ-lille3.fr/tata`.

[6] 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`.

[7] 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`.

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

[9] 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.

[10] 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.

[11] 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.

[12] 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.

[13] R. McNaughton & H. Yamada (1960): *Regular Expressions and State Graphs for Automata*. IEEE Trans. on Electronic Computers 9, pp. 39–47.

[14] Ludovic Mignot, Nadia Ouali Sebti & Djelloul Ziadi (2014): *An Efficient Algorithm for the Equation Tree Automaton via the k-C-Continuations*. In A. Beckmann, E. Csuhaj varjù & K. Meer (Eds.), editors: Computability in Europe- 10th International Conference, CiE 2014, Budapest, Hungary, June 23-27, 2014. Proceedings, Lecture Notes in Computer Science 8493, Springer, pp. 303–313.

[15] 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.