RAIRO-Theor. Inf. Appl. **42** (2007) 285–307
DOI: 10.1051/ita:2007035

Available online at:
www.rairo-ita.org

EFFICIENT WEIGHTED EXPRESSIONS CONVERSION

Faissal Ouardi[1] and Djelloul Ziadi[1]

Abstract. J. Hromkovič *et al.* have given an elegant method to convert a regular expression of size n into an ε-free nondeterministic finite automaton having $O(n)$ states and $O(n\log^2(n))$ transitions. This method has been implemented efficiently in $O(n\log^2(n))$ time by C. Hagenah and A. Muscholl. In this paper we extend this method to weighted regular expressions and we show that it can be achieved in $O(n\log^2(n))$ time.

Mathematics Subject Classification. 03D15, 68Q45.

Introduction

Weighted automata are efficient data structures for manipulating regular series. They are used in a lot of practical and theoretical applications such as computer algebra, image encoding, speech recognition or text processing. Regular series are also encoded by regular weighted expressions. The equivalence between these two representations, weighted regular expressions and finite automata has been proved in 1961 by Schützenberger [16]. For the conversion of weighted regular expressions into weighted automata there are principally three algorithms. The first one is due to Caron and Flouret [4]. Their algorithm works recursively on a subset of weighted regular expressions and produces the position automaton. Champarnaud *et al.* [7] have given an efficient algorithm that converts a weighted regular expression into its position automaton in quadratic time w.r.t. the size of the expression. Their algorithm is mainly based on the use of the $\mathbb{K}\mathcal{ZPC}$-structure for weighted expressions. The position automaton is a particular one (see [5]) that can have a quadratic number of transitions and $(n+1)$ states where n is the alphabetic width of the expression. The algorithm of Lombardy and Sakarovitch [14] constructs a weighted automaton that turns out to be the generalization of Antimirov automaton for the Boolean case. The Antimirov automaton [1] has

Keywords and phrases. Formal languages and automata, complexity of computation.

[1] L.I.T.I.S., University of Rouen, France; Faissal.Ouardi, Djelloul.Ziadi @univ-rouen.fr

Article published by EDP Sciences

less states than the position automaton. As the position automaton, Antimirov automaton can have a quadratic number of transitions.

In the Boolean case many algorithms have been developed for the problem of conversion [2,8,12,15,19]. The best one in term of complexity is that proposed by Hromkovič *et al.* [11] and implemented by Hagenah and Muscholl [9]. This automaton called the *common follow sets* automaton has $O(n)$ states and $O(n\log^2(n))$ transitions. Hagenah and Muscholl proved that this automaton can be constructed from a regular expression of size n in time $O(n\log^2(n))$ which constitutes the best known algorithm for the conversion problem.

In this paper, we extend the algorithm of Hromkovič *et al.* to weighted regular expressions and we present an efficient algorithm to convert a weighted regular expression of size n into an automaton having $O(n)$ states and $O(n\log^2(n))$ in $O(n\log^2(n))$ time.

In Section 1 we recall the notion of a $\mathbb{K}$-expression and of a formal series. In Section 2 we present the position automaton construction. In Section 3 we introduce the notion of *common follow polynomials* automaton which is the generalization of the notion of *common follow sets* automaton presented in [11]. In Section 4 we recall the $\mathbb{K}\mathcal{ZPC}$-structure in order to implement efficiently the algorithm introduced in Section 5. Finally in Section 6 and 7 we describe our algorithm.

1. Preliminaries

Let A be a finite alphabet, and $(\mathbb{K},\oplus,\otimes,0,1)$ be a semiring (commutative or not). The operator star $\circledast$ can be partially defined, the scalar $y = x^{\circledast} \in \mathbb{K}$ being a solution (if there exists) of the equations $y \otimes x \oplus 1 = y$ and $x \otimes y \oplus 1 = y$ with $0^{\circledast} = 1$ [10,13]. In this paper, we will give examples using the semiring $(\mathbb{Q},\oplus,\otimes)$ with the classical definitions for $\oplus$ (addition) and $\otimes$ (multiplication). In the following definition, we introduce the notion of $\mathbb{K}$-expression.

Definition 1.1. $\mathbb{K}$-*expressions* over an alphabet A are inductively defined as follows:

- $a \in A$ and $k \in \mathbb{K}$ are $\mathbb{K}$-expressions;
- if F and G are $\mathbb{K}$-expressions, then $(\mathrm{F}+\mathrm{G})$, $(\mathrm{F}\cdot\mathrm{G})$, and (F^*) are $\mathbb{K}$-expressions.

When there is no ambiguity, the $\mathbb{K}$-expression $(\mathrm{F}\cdot\mathrm{G})$ will be denoted $(\mathrm{F}\,\mathrm{G})$. Let E be a $\mathbb{K}$-expression. We will denote A_{E} the alphabet of E. The linearized version $\overline{\mathrm{E}}$ of E is the $\mathbb{K}$-expression deduced from E by ranking every letter occurrence with its position in E. Subscripted letters are called positions. The size of E, denoted $|\,\mathrm{E}\,|$ is the size of the syntax tree of E. For example, if $\mathrm{E} = (\frac{1}{2}\cdot a^* + \frac{1}{3}\cdot b^*)^*\cdot a^*$, we get $A_{\mathrm{E}} = \{a,b\}$, $\overline{\mathrm{E}} = (\frac{1}{2}\cdot a_1^* + \frac{1}{3}\cdot b_2^*)^*\cdot a_3^*$, $A_{\overline{\mathrm{E}}} = \{a_1,b_2,a_3\}$ and $|\overline{\mathrm{E}}| = 13$.

We define inductively the *null term* of a $\mathbb{K}$-expression E, denoted $c(\mathrm{E})$, by:

$$\begin{aligned} c(k) &= k \text{ for all } k \in \mathbb{K}, \\ c(a) &= 0 \text{ for all } a \in A, \\ c(\mathrm{F}+\mathrm{G}) &= c(\mathrm{F}) \oplus c(\mathrm{G}), \\ c(\mathrm{F}\,\mathrm{G}) &= c(\mathrm{F}) \otimes c(\mathrm{G}), \\ c(\mathrm{F}^*) &= c(\mathrm{F})^{\circledast}. \end{aligned}$$

The null term of $\mathrm{E} = (\frac{1}{2}a^* + \frac{1}{3}b^*)^*a^*$ is $c(\mathrm{E}) = 6$.

In the following, we present a brief description of formal series and we define a subset of the set of $\mathbb{K}$-expressions, usually called regular $\mathbb{K}$-expressions, which are associated to regular series [3].

Definition 1.2. A (non-commutative) formal series with coefficients in $\mathbb{K}$ and variables in A is a map from the free monoid A^* to $\mathbb{K}$ that associates with the word $w \in A^*$ a coefficient $\langle S, w\rangle \in \mathbb{K}$.

A formal series is usually written as an infinite sum: $S = \sum_{u\in A^*}\langle S, u\rangle u$. The *support* of the formal series S is the language $\mathrm{supp}(S) = \{u \in A^* \mid \langle S, u\rangle \neq 0\}$. The set of formal series over A with coefficients in $\mathbb{K}$ is denoted by $\mathbb{K}\langle\langle A\rangle\rangle$. A structure of semiring is defined on $\mathbb{K}\langle\langle A\rangle\rangle$ as follows [3,13]:

- $\langle S + T, u\rangle = \langle S, u\rangle \oplus \langle T, u\rangle$;
- $\langle ST, u\rangle = \bigoplus_{u_1u_2=u} \langle S, u_1\rangle \otimes \langle T, u_2\rangle$, with $S, T \in \mathbb{K}\langle\langle A\rangle\rangle$.

A polynomial is a formal series with finite support. The set of polynomials is denoted by $\mathbb{K}\langle A\rangle$. It is a subsemiring of $\mathbb{K}\langle\langle A\rangle\rangle$. The star of series is defined by: $S^* = \sum_{n\geq 0} S^n$ with $S^0 = \varepsilon$, $S^n = S^{n-1}S$ if $n > 0$. Notice that the star of a formal series does not always exist.

Proposition 1.3 (*cf.* [13])**.** *The star of a formal series $S \in \mathbb{K}\langle\langle A\rangle\rangle$ is defined if and only if $\langle S, \varepsilon\rangle^{\circledast}$ is defined in $\mathbb{K}$. In this case:*

$$S^* = \langle S, \varepsilon\rangle^{\circledast}(S_p\langle S, \varepsilon\rangle^{\circledast})^* \tag{1}$$

where the formal series S_p is defined by $\langle S_p, \varepsilon\rangle = 0$ and $\langle S_p, u\rangle = \langle S, u\rangle$ for any word u.

In the following, we will consider the previous construction of the star of formal series.

Definition 1.4. The semiring of regular series $\mathbb{K}Rat(A^*) \subset \mathbb{K}\langle\langle A\rangle\rangle$ is the smallest set of $\mathbb{K}\langle\langle A\rangle\rangle$ that contains the polynomials semiring $\mathbb{K}\langle A\rangle$, and that is stable by the operations of addition, product and star when this latter is defined.

The following definition introduces the notion of regular $\mathbb{K}$-expression.

Definition 1.5. A regular $\mathbb{K}$-expression is defined inductively by:

- $a \in A, k \in \mathbb{K}$ are regular $\mathbb{K}$-expressions that respectively denote the regular series $S_a = a$ and $S_k = k$;
- if F, G and H (such that $c(\mathrm{H})^{\circledast}$ exists) are regular $\mathbb{K}$-expressions that respectively denote the regular series S_{F}, S_{G} and S_{H}, then $(\mathrm{F} + \mathrm{G}), (\mathrm{F}\,\mathrm{G})$, and (H^*) are regular $\mathbb{K}$-expressions that respectively denote the regular series $S_{\mathrm{F}} + S_{\mathrm{G}}$, $S_{\mathrm{F}} S_{\mathrm{G}}$ and ${S_{\mathrm{H}}}^*$.

The set of regular $\mathbb{K}$-expressions over an alphabet A is denoted by $\mathbb{K}\,\mathrm{Exp}(A)$.

Definition 1.6. Let A be a finite alphabet, and $\mathbb{K}$ be a semiring (commutative or not). We define an automaton with multiplicities $\mathcal{A} = \langle Q, A, q_0, \delta, \gamma, \mu \rangle$ as follows:

- Q is a finite set of states;
- q_0 is the initial state;
- $\delta \subseteq Q \times A \times Q$ is the set of transitions;
- $\gamma \ : \ \delta \to \mathbb{K}$ is the transition weight function;
- $\mu \ : \ Q \to \mathbb{K}$ is the output weight function.

The definition of an automaton with multiplicities is more general but this one is sufficient for our construction.

A path p from a state q to a state q' is a sequence of transitions (q, a_1, q_1), $(q_1, a_2, q_2), \cdots, (q_{n-1}, a_n, q_n = q')$ in $\mathcal{A}$. It is written $p = (p_1, p_2, \cdots, p_n)$ with $p_i = (q_{i-1}, a_i, q_i)$ for $1 \leq i \leq n$. Its label is the word $w(p) = a_1 a_2 \cdots a_n$. We denote by $\mathrm{coef}(p)$ the cost of the path p in $\mathcal{A}$. Formally, $\mathrm{coef}(p) = \gamma(p_1) \otimes \gamma(p_2) \otimes \cdots \otimes \gamma(p_n) \otimes \mu(q_n)$.

Let $\mathcal{C}_{\mathcal{A}}$ the set of all paths in $\mathcal{A}$ starting at q_0. The series $S_{\mathcal{A}}$ associated with the automaton $\mathcal{A}$ is

$$S_{\mathcal{A}} = \sum_{u \in A^*} \langle S_{\mathcal{A}}, u \rangle u$$

where:

$$\langle S_{\mathcal{A}}, u \rangle = \bigoplus_{\substack{p \in \mathcal{C}_{\mathcal{A}} \\ w(p) = u}} \mathrm{coef}(p).$$

We say that the automaton $\mathcal{A}$ *realizes* the series $S_{\mathcal{A}}$.

Definition 1.7. A formal series $S \in \mathbb{K}\langle\langle A \rangle\rangle$ is called recognizable if there exists an automaton that realizes it.

The following result due to Schützenberger [16] is classical.

Theorem 1.8. *(Schützenberger, 1961). A formal series is recognizable if and only if it is regular.*

Let S be a series in $\mathbb{K}\langle\langle A \rangle\rangle$. If $h : A \longrightarrow B$ is a surjective morphism, then $h(S)$ denotes the series $\sum_{u \in A^*} \langle S, u \rangle h(u)$ in $\mathbb{K}\langle\langle A \rangle\rangle$.

Proposition 1.9. *Let $\mathcal{A} = \langle Q, X, q_0, \delta, \gamma, \mu\rangle$ and $\mathcal{B} = \langle Q, Y, q_0, \delta', \gamma', \mu\rangle$ be two automata. Let $h\colon X \to Y$ be a surjective morphism where*

(1) $(p, a, q) \in \delta' \Leftrightarrow$ *There exists* $(p, a_i, q) \in \delta$ *such that* $h(a_i) = a$,
(2) $\gamma'(p, a, q) = \bigoplus\limits_{\substack{h(a_i)=a \\ (p,a_i,q)\in\delta}} \gamma(p, a_i, q)$.

We have, $S_{\mathcal{B}} = h(S_{\mathcal{A}})$.

Proof. Let $\Omega_{\mathcal{A}}$ be the set of sequences in $\mathcal{A}$,

$$\Omega_{\mathcal{A}} = \{(q_0, q_1), (q_1, q_2), \cdots, (q_{n-1}, q_n) | \exists a_{k_i} \in X, \text{ s.t. } (q_{i-1}, a_{k_i}, q_i) \in \delta, \forall 1 \le i \le n\}. \quad (2)$$

We define the transitions polynomial between two states q and q' as

$$\Delta_{\mathcal{A}}(q, q') = \sum_{(q,a,q')\in\delta} \gamma(q, a, q')a.$$

For a sequence $\rho = ((q_0, q_1), (q_1, q_2), \cdots, (q_{n-1}, q_n))$ in $\Omega_{\mathcal{A}}$, we set

$$\Delta_{\mathcal{A}}(\rho) = \bigotimes_{i=1}^{n} \Delta_{\mathcal{A}}(q_{i-1}, q_i)\mu(q_n).$$

Thus, the series $S_{\mathcal{A}}$ associated with the automaton $\mathcal{A}$ can be written as

$$S_{\mathcal{A}} = \sum_{\rho\in\Omega_{\mathcal{A}}} \Delta_{\mathcal{A}}(\rho).$$

Now let us prove that $h(S_{\mathcal{A}}) = S_{\mathcal{B}}$. We have

$$\begin{aligned} h(S_{\mathcal{A}}) &= h\left(\sum_{\rho\in\Omega_{\mathcal{A}}} \Delta_{\mathcal{A}}(\rho)\right) \\ &= h\left(\sum_{\rho=((q_0,q_1),(q_1,q_2),\cdots,(q_{n-1},q_n))\in\Omega_{\mathcal{A}}} \left(\bigotimes_{i=1}^{n} \Delta_{\mathcal{A}}(q_{i-1}, q_i)\right)\right)\mu(q_n) \\ &= \sum_{\rho\in\Omega_{\mathcal{A}}} \bigotimes_{i=1}^{n} \left[\sum_{(q_{i-1},a_i,q_i)\in\delta} \gamma(q_{i-1}, a_i, q_i)h(a_i)\mu(q_n)\right] \end{aligned}$$

$$
\begin{aligned}
&\overset{Condition\ (1)}{=} && \sum_{\rho\in\Omega_{\mathcal{A}}} \bigotimes_{i=1}^{n} \sum_{(q_{i-1},a,q_i)\in\delta'} \left[\left(\bigoplus_{\substack{(q_{i-1},a_{k_i},q_i)\in\delta \\ h(a_{k_i})=a}} \gamma(q_{i-1},a_{k_i},q_i) \right) a\mu(q_n) \right] \\
&\overset{Condition\,(2)}{=} && \sum_{\rho\in\Omega_{\mathcal{A}}} \bigotimes_{i=1}^{n} \sum_{(q_{i-1},a,q_i)\in\delta'} [\gamma'(q_{i-1},a,q_i) a\mu(q_n)] \\
&= && \sum_{\rho\in\Omega_{\mathcal{A}}} \Delta_{\mathcal{B}}(\rho) \\
&\overset{Condition\ (1),Relation\ 2}{=} && \sum_{\rho\in\Omega_{\mathcal{B}}} \Delta_{\mathcal{B}}(\rho) \\
&= && S_{\mathcal{B}}. \qquad \square
\end{aligned}
$$

2. Position automaton

In this section, we recall some basic notions related to the construction of position automata from regular $\mathbb{K}$-expressions. Let E be a regular $\mathbb{K}$-expression over an alphabet A. The position automaton associated with E is computed from three functions First(E), Last(E) and Follow$(\cdot,\mathrm{E})$ in $\mathbb{K}\,\mathrm{Exp}(A) \longrightarrow \mathbb{K}\langle A\rangle$. They can be computed inductively according to the following rules [7]:

$$
\begin{aligned}
\mathrm{First}(k) &= 0 \text{ for all } k\in\mathbb{K}, && (3)\\
\mathrm{First}(a) &= 1a_i \ (a_i \text{ is the position associated to } a \text{ in } A_{\overline{\mathrm{E}}}) , && (4)\\
\mathrm{First}(\mathrm{F}+\mathrm{G}) &= \mathrm{First}(\mathrm{F}) + \mathrm{First}(\mathrm{G}), && (5)\\
\mathrm{First}(\mathrm{F}\,\mathrm{G}) &= \mathrm{First}(\mathrm{F}) + c(\mathrm{F})\,\mathrm{First}(\mathrm{G}), && (6)\\
\mathrm{First}(\mathrm{F}^*) &= c(\mathrm{F})^{\circledast}\,\mathrm{First}(\mathrm{F}). && (7)
\end{aligned}
$$

We obtain the same rule for Last by substituting Last for First and replacing the formula (6) by:

$$
\mathrm{Last}(\mathrm{F}\,\mathrm{G}) = \mathrm{Last}(\mathrm{G}) + c(\mathrm{G})\,\mathrm{Last}(\mathrm{F}). \tag{8}
$$

Given a position $x\in A_{\overline{\mathrm{E}}}$, the function Follow$(x,\mathrm{E})$ is inductively computed as follows:

$$
\begin{aligned}
\mathrm{Follow}(x,k) &= 0 \text{ for all } k\in\mathbb{K},\\
\mathrm{Follow}(x,a) &= 0 \text{ for all } a\in A,\\
\mathrm{Follow}(x,\mathrm{F}+\mathrm{G}) &= \mathrm{Follow}(x,\mathrm{F}) + \mathrm{Follow}(x,\mathrm{G}),\\
\mathrm{Follow}(x,\mathrm{F}\,\mathrm{G}) &= \mathrm{Follow}(x,\mathrm{F}) + \langle \mathrm{Last}(\mathrm{F}),x\rangle\,\mathrm{First}(\mathrm{G}) + \mathrm{Follow}(x,\mathrm{G}),\\
\mathrm{Follow}(x,\mathrm{F}^*) &= \mathrm{Follow}(x,\mathrm{F}) + \langle \mathrm{Last}(\mathrm{F}^*),x\rangle\,\mathrm{First}(\mathrm{F}).
\end{aligned}
$$

Let h be the mapping from $A_{\overline{\mathrm{E}}}$ to A_{E} induced by the linearization of E over $A_{\overline{\mathrm{E}}}$. It maps every position to its value in A_{E}. For example, if $\overline{\mathrm{E}} = 2a_1 + 3b_2 + a_3$ then $h(a_1) = h(a_3) = a$ and $h(b_2) = b$. The First, Last and Follow polynomials of a regular $\mathbb{K}$-expression E can be used to define an automaton with multiplicities realizing S_{E}. We define the *position automaton* for E, denoted $\mathcal{A}_{\mathrm{E}}$, as the 6-tuple $\langle Q, A_{\mathrm{E}}, q_0, \delta, \gamma, \mu\rangle$ where, for some $q_0 \notin A_{\overline{\mathrm{E}}}$:

- $Q = \{q_0\} \cup A_{\overline{\mathrm{E}}}$.
- $(q, a, p) \in \delta \Leftrightarrow h(p) = a$ and if $q = q_0$ then $\langle \mathrm{First}(\mathrm{E}), p\rangle \neq 0$, else $\langle \mathrm{Follow}(q, \mathrm{E}), p\rangle \neq 0$.
- For all $(p, a, q) \in \delta$, one has:
  $$\gamma(q, a, p) = \begin{cases} \langle \mathrm{First}(\mathrm{E}), p\rangle & \text{if } q = q_0, \\ \langle \mathrm{Follow}(q, \mathrm{E}), p\rangle & \text{otherwise.} \end{cases}$$
- $$\mu(q) = \begin{cases} c(\mathrm{E}) & \text{if } q = q_0, \\ \langle \mathrm{Last}(\mathrm{E}), q\rangle & \text{otherwise.} \end{cases}$$

Proposition 2.1. *[7] Let* E *be a regular* $\mathbb{K}$*-expression and* $\mathcal{A}_{\mathrm{E}}$ *its position automaton. Then* $\mathcal{A}_{\mathrm{E}}$ *realizes the regular series* S_{E}*.*

Lemma 2.2. *Let* E *be a regular* $\mathbb{K}$*-expression and* $\overline{\mathrm{E}}$ *its linearized version. Then one has* $S_{\overline{\mathrm{E}}} = h(S_{\mathrm{E}})$*.*

Proof. Let $\mathcal{A}_{\mathrm{E}}$ be the position automaton associated with E and $\mathcal{A}_{\overline{\mathrm{E}}}$ be the position automaton associated with $\overline{\mathrm{E}}$. The surjective morphism h from $A_{\overline{\mathrm{E}}}$ into A_{E} satisfies the conditions (1) and (2) of Proposition 1.9. Then, one has:

$$S_{\mathcal{A}_{\mathrm{E}}} \quad = \quad h(S_{\mathcal{A}_{\overline{\mathrm{E}}}}).$$

From Proposition 2.1, we have $S_{\mathcal{A}_{\mathrm{E}}} = S_{\mathrm{E}}$ and $S_{\mathcal{A}_{\overline{\mathrm{E}}}} = S_{\overline{\mathrm{E}}}$. Thus $S_{\mathrm{E}} = h(S_{\overline{\mathrm{E}}})$. □

3. Common follow polynomials automaton

In this section, we introduce the notion of common follow polynomials system which is the generalization of the notion of the common follow sets introduced by Hromkovič *et al.* [11] in order to compute an automaton having less transitions than the position automaton.

Next, we prove that the CFP automaton induced by the *common follow polynomials* system realizes the same series as the position automaton.

Finally, we recall the $\mathbb{K}\mathcal{ZPC}$-structure in order to implement an efficient algorithm to convert a regular $\mathbb{K}$-expression into its CFP automaton.

From now, we will consider the expression $\mathrm{E}' = (\overline{\mathrm{E}} \cdot \sharp)$ with $\sharp \notin A_{\overline{\mathrm{E}}}$.

Definition 3.1. Let E be a regular $\mathbb{K}$-expression and let x be a position in $A_{\overline{\mathrm{E}}}$. The set $\mathrm{dec}(x) = \{(k_1, P_1), \cdots, (k_m, P_m)\}$ where $\{P_1, \cdots P_m\}$ is a set of polynomials with pairwise disjoints supports and $\{k_1, \cdots, k_m\}$ m elements of $\mathbb{K}$, is called *follow decomposition of* x *if and only if one has:*

$$\mathrm{Follow}(x, \mathrm{E}) = k_1P_1 + k_2P_2 + \cdots + k_mP_m.$$

Proposition 3.2. *Let x be a position in $A_{\overline{\mathrm{E}}}$, and let* $\mathrm{dec}(x) = \{(k_1, P_1), \cdots, (k_m, P_m)\}$ *and* $\mathrm{dec}'(x) = \{(k'_1, P_1), \cdots, (k'_m, P_m)\}$ *be two follow decompositions of x. Then, one has $k_i = k'_i$, for all $1 \leq i \leq m$.*

Proof. It comes from the fact that $\mathrm{supp}(P_i) \cap \mathrm{supp}(P_j) = \emptyset$, for all $1 \leq i, j \leq m$ and $i \neq j$. □

We will write $k_x^{P_i}$ the coefficient of P_i in the follow decomposition of x and $\Pr(\mathrm{dec}(x))$ the set $\{P_1, \cdots, P_m\}$.

Definition 3.3 (*common follow polynomials system*)**.** Let E be a regular $\mathbb{K}$-expression and $\mathrm{E}' = \overline{\mathrm{E}}\sharp$. A *common follow polynomials system* $S(\mathrm{E}) = (\mathrm{dec}(x))_{x \in A_{\overline{\mathrm{E}}}}$ of E is a family of follow decompositions of positions in $A_{\overline{\mathrm{E}}}$. $\mathcal{F}_{S(\mathrm{E})} = \{\mathrm{First}(\mathrm{E}')\} \cup \bigcup_{x \in A_{\overline{\mathrm{E}}}} \Pr(\mathrm{dec}(x))$ is called the set of *common follow polynomials* associated with the system $S(E)$.

Notice that the *common follow polynomial system* $S(\mathrm{E})$ is not unique as we can see in the following example.

Example 3.4. Let $\mathrm{E} = (\frac{1}{3}a + \frac{1}{6}b)(\frac{1}{2}a^*b)$. One has:

$$\begin{aligned}
\mathrm{E}' &= (\frac{1}{3}a_1 + \frac{1}{6}b_2)(\frac{1}{2}a_3^*b_4)\sharp,\\
\mathrm{First}(\mathrm{E}') &= \frac{1}{3}a_1 + \frac{1}{6}b_2,\\
\mathrm{Follow}(a_1, \mathrm{E}') &= \frac{1}{2}a_3 + \frac{1}{2}b_4,\\
\mathrm{Follow}(b_2, \mathrm{E}') &= \frac{1}{2}a_3 + \frac{1}{2}b_4,\\
\mathrm{Follow}(a_3, \mathrm{E}') &= a_3 + b_4,\\
\mathrm{Follow}(b_4, \mathrm{E}') &= \sharp.
\end{aligned}$$

$S(\mathrm{E}) = \{$ $\mathrm{dec}(a_1) = \{(\frac{1}{2}, (a_3 + b_4))\}$, $\mathrm{dec}(b_2) = \{(\frac{1}{2}, (a_3 + b_4))\}$, $\mathrm{dec}(a_3) = \{(1, (a_3 + b_4))\}$, $\mathrm{dec}(b_4) = \{(1, \sharp)\}$ $\}$	$S'(\mathrm{E}) = \{$ $\mathrm{dec}(a_1) = \{(\frac{1}{2}, a_3), (\frac{1}{2}, b_4)\}$, $\mathrm{dec}(b_2) = \{(\frac{1}{2}, a_3), (\frac{1}{2}, b_4)\}$, $\mathrm{dec}(a_3) = \{(1, (a_3 + b_4))\}$, $\mathrm{dec}(b_4) = \{(1, \sharp)\}$ $\}$
$\mathcal{F}_{S(\mathrm{E})} = \{\frac{1}{3}a_1 + \frac{1}{6}b_2, a_3 + b_4, \sharp\}$	$\mathcal{F}_{S'(\mathrm{E})} = \{\frac{1}{3}a_1 + \frac{1}{6}b_2, a_3, b_4, a_3 + b_4, \sharp\}$

Definition 3.5 (CFP automaton)**.** Let E a regular $\mathbb{K}$-expression. Let $S(\mathrm{E})$ be a *common follow polynomials system* of E. The *common follow polynomials automaton* $\mathcal{A}_{S(\mathrm{E})} = \langle Q, A_{\mathrm{E}}, q_0, \delta_1, \gamma_1, \mu \rangle$ associated with $S(\mathrm{E})$ is defined by:

- $Q = \mathcal{F}_{S(\mathrm{E})}$,
- $q_0 = \{\mathrm{First}(\mathrm{E}')\}$,
- $(P, a, P') \in \delta_1 \Leftrightarrow$ There exists $a_i \in A_{\overline{\mathrm{E}}}$ such that the following holds:

  - $h(a_i) = a$,
  - $\langle P, a_i\rangle \neq 0$,
  - $P' \in \mathrm{Pr}(\mathrm{dec}(a_i))$.
- For all $(P, a, P') \in \delta_1$, one has: $\gamma_1(P, a, P') = \bigoplus_{\substack{a_i \in A_{\overline{\mathrm{E}}} \\ h(a_i)=a}} \langle P, a_i\rangle \otimes k_{a_i}^{P'}$.
- $\mu(P) = \langle P, \sharp\rangle$.

Theorem 3.6. *Let* E *be a regular* $\mathbb{K}$*-expression and* $\mathcal{A}_{S(\mathrm{E})}$ *the common follow polynomials automaton associated with a CFP system* $S(\mathrm{E})$*. Then* $\mathcal{A}_{S(\mathrm{E})}$ *realizes the regular series* S_{E}*.*

To prove this theorem, we introduce the automaton $\mathcal{A}_{S(\overline{\mathrm{E}})} = \langle Q, A_{\overline{\mathrm{E}}}, q_0, \delta_2, \gamma_2, \mu\rangle$ defined as follows:

- $Q = \mathcal{F}_{S(\mathrm{E})}$.
- $q_0 = \{\mathrm{First}(\mathrm{E}')\}$.
- $(P, a_i, P') \in \delta_2 \Leftrightarrow$ The following holds:
  - $\langle P, a_i\rangle \neq 0$,
  - $P' \in \mathrm{Pr}(\mathrm{dec}(a_i))$.
- For all $(P, a_i, P') \in \delta_2$, one has: $\gamma_2(P, a_i, P') = \langle P, a_i\rangle \otimes k_{a_i}^{P'}$.
- $\mu(P) = \langle P, \sharp\rangle$.

Lemma 3.7. *Let* E *be a regular* $\mathbb{K}$*-expression. The automaton* $\mathcal{A}_{S(\overline{\mathrm{E}})}$ *realizes the series* $S_{\overline{\mathrm{E}}}$*.*

Proof. Let E be a regular $\mathbb{K}$-expression and $\mathcal{A}_{\overline{\mathrm{E}}}$ the position automaton associated to its linearized version $\overline{\mathrm{E}}$ and let $S(\mathrm{E}) = (\mathrm{dec}(x))_{x \in A_{\overline{\mathrm{E}}}}$ a CFP system associated with E. Let $w = a_1 a_2 \cdots a_n \in A_{\overline{\mathrm{E}}}^*$, From Proposition 2.1, $\mathcal{A}_{\overline{\mathrm{E}}}$ realizes the series $S_{\overline{\mathrm{E}}}$, so to prove this lemma it suffices to show that for each path $p \in \mathcal{C}_{\mathcal{A}_{\overline{\mathrm{E}}}}$, such that $m(p) = w$, there exists a unique equivalent path $p' \in \mathcal{C}_{\mathcal{A}_{S(\overline{\mathrm{E}})}}$ such that $m(p') = w$ and $\mathrm{coef}(p) = \mathrm{coef}(p')$, and conversely.

Let $p = (q_0, a_1, q_1), \cdots, (q_{n-1}, a_n, q_n)$ be a path in $\mathcal{C}_{\mathcal{A}_{\overline{\mathrm{E}}}}$. According to the definition of the automaton $\mathcal{A}_{\overline{\mathrm{E}}}$, one has:

$$\langle \mathrm{First}(\mathrm{E}), a_1\rangle \neq 0, \tag{9}$$

$$\langle \mathrm{Follow}(a_i, \mathrm{E}), a_{i+1}\rangle \neq 0, \ \forall 1 \leq i \leq n-1. \tag{10}$$

Let P_i the unique polynomial in $\mathrm{Pr}(\mathrm{dec}(a_i))$ such that $\langle P_i, a_{i+1}\rangle \neq 0$ *i.e.* $a_{i+1} \in \mathrm{supp}(P_i)$ and $P_i \cap P_j = \emptyset \ \forall 1 \leq i \neq j \leq n$. P_i exists since $\langle \mathrm{Follow}(a_i, \mathrm{E}), a_{i+1}\rangle \neq 0$. So the path

$$p' = (\mathrm{First}(\mathrm{E}), a_1, P_1), (P_1, a_2, P_2), \cdots, (P_{n-1}, a_n, P_n) \tag{11}$$

exists in $\mathcal{A}_{S(\overline{\mathrm{E}})}$ and is unique.

Let us prove that $\mathrm{coef}(p) = \mathrm{coef}(p')$. One has:

$$\begin{array}{rcl} \mathrm{coef}(p) & = & \gamma(q_0, a_1, q_1) \otimes \gamma(p_1, a_2, q_2) \cdots \gamma(q_{n-1}, a_n, q_n) \otimes \mu(q_n), \\ & \overset{Def.\ of\ \mathcal{A}_{\overline{\mathrm{E}}}}{=} & \langle \mathrm{First(E)}, a_1 \rangle \otimes \langle \mathrm{Follow}(a_1, E), a_2 \rangle \cdots \langle \mathrm{Follow}(a_{n-1}, E), a_n \rangle \\ & & \otimes \langle \mathrm{Last(E)}, a_n \rangle; \end{array}$$

and

$$\begin{array}{rcl} \mathrm{coef}(p') & = & \gamma_2(\mathrm{First(E)}, a_1, P_1) \otimes \gamma_2(P_1, a_2, P_2) \otimes \cdots \otimes \\ & & \gamma_2(P_{n-1}, a_n, P_n) \otimes \mu(P_n), \\ & \overset{Def.\ of\ \mathcal{A}_{S(\overline{\mathrm{E}})}}{=} & \langle \mathrm{First(E)}, a_1 \rangle \otimes k_{a_1}^{P_1} \otimes \langle P_1, a_2 \rangle \otimes k_{a_2}^{P_2} \cdots \langle P_{n-1}, a_n \rangle \\ & & \otimes k_{a_n}^{P_{n-1}} \otimes \mu(p_n). \end{array}$$

From Definition 3.1, We have $\mathrm{Follow}(a_i, \mathrm{E}) = k_{a_i}^{P_{i_1}} P_{i_1} + \cdots + k_{a_i}^{P_i} P_i + \cdots + k_{a_i}^{P_{i_n}} P_{i_n}$. Then, one has
$k_{a_i}^{P_i} \otimes \langle P_i, a_{i+1} \rangle = \langle \mathrm{Follow}(a_i, \mathrm{E}), a_{i+1} \rangle$, for all $1 \leq i \leq n-1$ and $\langle \mathrm{Last(E)}, a_n \rangle = \mu(P_n)$. □

Proof of Theorem 3.6. Let E be a regular $\mathbb{K}$-expression. Let h be the surjective morphism from $A_{\overline{\mathrm{E}}}$ into A_{E}. We consider the automata $\mathcal{A}_{S(\overline{\mathrm{E}})}$ and $\mathcal{A}_{S(\mathrm{E})}$, the application h satisfies the conditions (1) and (2) of Proposition 1.9, then one has:

$$\begin{array}{rcl} S_{\mathcal{A}_{S(\mathrm{E})}} & = & h(S_{\mathcal{A}_{S(\overline{\mathrm{E}})}}), \\ & \overset{Lemma\ 3.7}{=} & h(S_{\overline{\mathrm{E}}}), \\ & \overset{Lemma\ 2.2}{=} & S_{\mathrm{E}}. \end{array}$$ □

Example 3.8. Consider the regular $\mathbb{K}$-expression $\mathrm{E} = (\frac{1}{3}a + \frac{1}{6}b + \frac{1}{2}a)(\frac{1}{2}a^*b)$. The linearized version of E is $\overline{\mathrm{E}} = (\frac{1}{3}a_1 + \frac{1}{6}b_2 + \frac{1}{2}a_3)(\frac{1}{2}a_4^*b_5)$. Consider the following set of *common follow polynomials* $\mathcal{F}_{S(\mathrm{E})} = \{\frac{1}{3}a_1 + \frac{1}{6}b_2 + \frac{1}{2}a_3\} \cup \{a_4 + b_5, \sharp\}$.

Note that for the previous example, the position automaton associated with E has 6 states and 11 transitions (see Fig. 1). However there exists a CFP automaton realizing the series S_{E} with only 3 states and 4 transitions (see Fig. 2).

The number of transitions and the number of states in a *common follow polynomials automaton* obviously depends on the choice of the *common follow polynomials system*. The next section deals with the problem of finding appropriate common follow polynomials system.
A good choice can be found by resolving the following system:

$$\left\{ \begin{array}{rcl} Minimize(f(\mathrm{E}) & = & \displaystyle\sum_{x \in A_{\overline{\mathrm{E}}}} n_x m_x) \end{array} \right.$$

where m_x denotes the number of polynomials $C \in \mathcal{F}_{S(\mathrm{E})}$ such that $\langle C, x \rangle \neq 0$ and n_x denotes the size of $\mathrm{dec}(x)$.

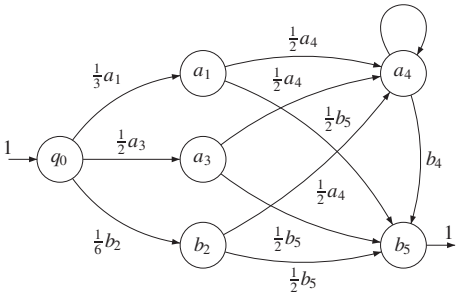

FIGURE 1. The position automaton for $\overline{\mathrm{E}} = (\frac{1}{3}a_1 + \frac{1}{6}b_2 + \frac{1}{2}a_3)(\frac{1}{2}a_4^* b_5)$.

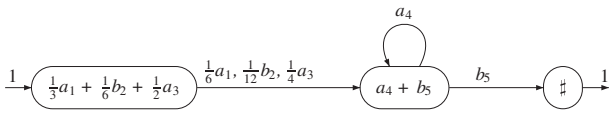

FIGURE 2. A CFP automaton for $\overline{\mathrm{E}} = (\frac{1}{3}a_1 + \frac{1}{6}b_2 + \frac{1}{2}a_3)(\frac{1}{2}a_4^* b_5)$.

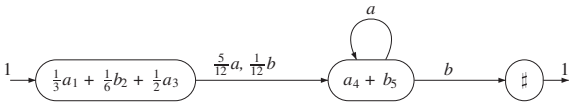

FIGURE 3. A CFP automaton for $\mathrm{E} = (\frac{1}{3}a + \frac{1}{6}b + \frac{1}{2}a)(\frac{1}{2}a^* b)$.

From the definition of the automaton $\mathcal{A}_{S(\overline{\mathrm{E}})}$, the function f represents the number of its transitions.

Unfortunately, in the practice this method is not efficient. In the Boolean case J. Hromkovič *et al.* [11], presented an elegant method that computes a particular *common follow sets system* which yields to a *common follow sets automaton* having $O(n)$ states and $O(n\log^2(n))$ transitions where n denote the size of the regular expression. In [9] C. Hagenah and A. Muscholl have shown that this particular automaton can be computed in time $O(n\log^2(n))$ which is the best known algorithm for the problem of conversion in the boolean case.

In the next sections, we prove that for a regular $\mathbb{K}$-expression of size n there exists a *common follow polynomials automaton* with $O(n\log^2(n))$ transitions and $O(n)$ states. This constitutes the generalization of the result proved by J. Hromkovič *et al.* [11]. Next we give an efficient algorithm that computes this automaton in time $O(n\log^2(n))$.

Our algorithm is based on the $\mathbb{K}\mathcal{ZPC}$-structure . This structure has been introduced in [7], in order to compute the position automaton. Its boolean version is described in [17,19]. The following section gives a brief description of this structure.

4. $\mathbb{K}\mathcal{ZPC}$-STRUCTURE

Let E be a regular $\mathbb{K}$-expression. The $\mathbb{K}\mathcal{ZPC}$-structure of E is based on two labeled trees (TL(E) and TF(E)) deduced from its syntax tree $\mathrm{T}(\overline{\mathrm{E}})$. These trees encode respectively the Last and the First polynomials associated to the subexpressions of E. The edges of these trees are labeled by elements of the semiring $\mathbb{K}$.

A node in $\mathrm{T}(\overline{\mathrm{E}})$ will be noted ν. If the arity of ν is two, we write respectively ν_l and ν_r its left son and its right son. If its arity is 1, its son will be noted ν_s. The relation of descendance over the syntax tree is denoted $\preceq$. Let (ν_1, ν_2) an edge in TL(E) (or in TF(E)), we denote by $\alpha(\nu_1, \nu_2)$ its label. For a tree whose edges are labeled by elements of the semiring $\mathbb{K}$, we define the cost of the path $(\nu = \nu_1, \nu_2, \cdots, \nu_k = \nu')$, written $\pi(\nu, \nu')$, as

$$\bigotimes_{i=1}^{k-1} \alpha(\nu_i, \nu_{i+1}).$$

By convention we set $\pi(\nu, \nu) = 1$.

A subtree t of $\mathrm{T}(\overline{\mathrm{E}})$ is a tree associated to a subexpression of $\overline{\mathrm{E}}$, or a tree obtained from $\mathrm{T}(\overline{\mathrm{E}})$ by deleting a set of trees which represent some subexpressions of $\overline{\mathrm{E}}$. This definition is also applied to t. Let t_1 be a subtree of t, we denote by $\mathrm{t} \setminus \mathrm{t}_1$ the subtree of t resulting from t by deleting the subtree t_1. We denote by Pos(t) the set of positions of $A_{\overline{\mathrm{E}}}$ in t. As a measure of t we use the cardinality of the nodes set of t. It denoted by $|\,\mathrm{t}\,|$. For a node ν in T(E), the regular $\mathbb{K}$-expression E_ν denotes the subexpression resulting from the node ν and $c(\nu)$ its null term. The Last tree TL(E) is a labeled copy of $\mathrm{T}(\overline{\mathrm{E}})$, where an edge going from a node λ labeled "$\cdot$" to its left son λ_l is marked by $\alpha(\lambda_l, \lambda) = c(\lambda_l)$ and for each edge going from a node λ labeled "$*$" to its son λ_s is marked by $\alpha(\lambda_s, \lambda) = c(\lambda_s)^{\circledast}$. For all other edges (λ, λ'), we set $\alpha(\lambda, \lambda') = 1$. The node λ represents the polynomial

$$e(\lambda) \quad = \quad \sum_{x \in A_{\overline{\mathrm{E}}}} \pi(x, \lambda) x. \tag{12}$$

Thus we have

$$e(\lambda) \quad = \quad \mathrm{Last}(\mathrm{E}_\lambda).$$

The First tree TF(E) is computed in a similar way, by marking an edge going from a node φ labeled "$\cdot$" to its right son φ_r by $\alpha(\varphi, \varphi_r) = c(\varphi_l)$ and a edge going from a node φ to its son φ_s is marked by $\alpha(\varphi_s, \varphi) = c(\varphi_s)^{\circledast}$. For all other edges (φ, φ'), we set $\alpha(\varphi, \varphi') = 1$. The node φ represents the polynomial

$$e(\varphi) \quad = \quad \sum_{x \in A_{\overline{\mathrm{E}}}} \pi(\varphi, x) x. \tag{13}$$

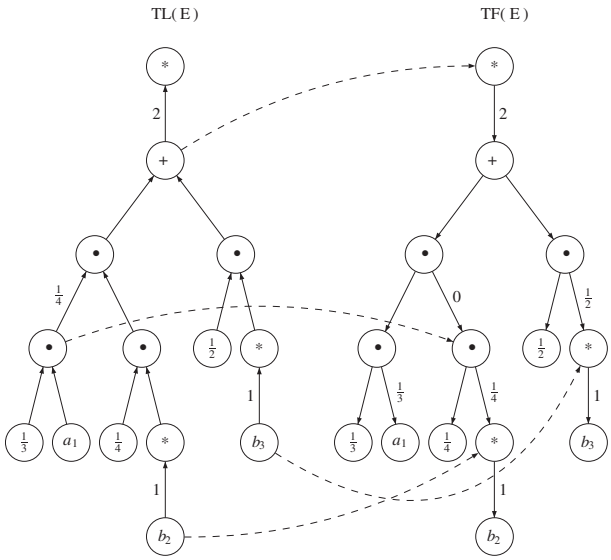

FIGURE 4. The $\mathbb{K}\mathcal{ZPC}$-structure associated to the expression $\mathrm{E} = (\frac{1}{3}a\frac{1}{4}b^* + \frac{1}{2}b^*)^*$.

Thus we have

$$e(\varphi) \quad = \quad \mathrm{First}(\mathrm{E}_\varphi).$$

The two trees are connected as follows: if a node λ of TL(E) is labeled by "$\cdot$", its left son λ_l is linked to the right son φ_r of the corresponding node φ in TF(E). If a node in TL(E) labeled "$*$" its son node is linked to its corresponding node in TF(E). Such links are called *follow links*. The set of follow links is denoted by Δ. We denote by Δ_x the set of follow links associated to the position x. That is: $\Delta_x = \{(\lambda, \varphi) \in \Delta \mid x \preceq \lambda\}$.

Proposition 4.1. *Let* E *be a regular* $\mathbb{K}$*-expression and* $x \in A_{\overline{\mathrm{E}}}$*. Then*

$$\mathrm{Follow}(x, \mathrm{E}) \quad = \quad \sum_{(\lambda,\varphi)\in\Delta_x} \pi(x, \lambda)e(\varphi). \tag{14}$$

5. A CFP system computation

Now we describe a procedure which allows us to produce a CFP system that yields to a CFP automaton with $O(n\log^2(n))$ transitions and $O(n)$ states. This procedure is a generalization of J. Hromkovič *et al.* one.

We introduce the function $f : \mathrm{TL}(\mathrm{E}') \to \mathrm{TF}(\mathrm{E}') \cup \{\perp\}$ defined by:

$$f(\lambda) = \begin{cases} \varphi & \text{if } (\lambda, \varphi) \in \Delta_x, \\ \perp & \text{otherwise,} \end{cases}$$

where $\perp$ denotes an artificial node such that $\pi(x, \perp) = 0$.
We extend the polynomial Follow to the nodes of the tree $\mathrm{TL}(\mathrm{E}')$ and $\mathrm{TF}(\mathrm{E}')$ as follows:
if λ_1 and λ_2 are two nodes in $\mathrm{TL}(\mathrm{E}')$ such that $\lambda_1 \preceq \lambda_2$, then

$$\mathrm{Follow}(\lambda_1, \lambda_2) = \sum_{\substack{\lambda_1 \preceq \lambda \prec \lambda_2 \\ f(\lambda) = \varphi}} \pi(\lambda_1, \lambda) e(\varphi), \tag{15}$$

and if φ_1 and φ_2 are two nodes in $\mathrm{TF}(\mathrm{E}')$ such that $\varphi_1 \preceq \varphi_2$, then

$$\mathrm{Follow}(\varphi_1, \varphi_2) = \sum_{\substack{\varphi_1 \preceq \varphi \prec \varphi_2 \\ f(\lambda) = \varphi}} \pi(\varphi_1, \varphi) e(\lambda). \tag{16}$$

Let P be a polynomial and let t be a subtree of $\mathrm{T}(\mathrm{E}')$. The restriction of P to t, denoted by P_{t} is the polynomial

$$P_{\mathrm{t}} = \sum_{x \in \mathrm{Pos(t)}} \langle P, x \rangle x.$$

Let t be a subtree of $\mathrm{T}(\mathrm{E}')$ and x a position in $\mathrm{Pos(t)} \setminus \{\sharp\}$. We denote by tf (respectively tl) the labeled copy of t in $\mathrm{TF}(\mathrm{E}')$ (respectively in $\mathrm{TL}(\mathrm{E}')$). The procedure recursively computes a particular CFP system. It is defined as follows.

If $|\,\mathrm{Pos(t)}| > 1$
Decompose t into two subtrees t_1 and t_2 according to the following rules:
$\frac{|\,\mathrm{Pos(t)}|}{3} \leq |\,\mathrm{Pos}(\mathrm{t}_1)| \leq \frac{2|\,\mathrm{Pos(t)}|}{3}$ and let $\mathrm{t}_2 = \mathrm{t} \setminus \mathrm{t}_1$ with λ_1 be the root of the subtree tl_1 and φ_1 its corresponding node in tf_1. Let

$$\begin{aligned} P_1 &= \mathrm{Follow}_{\mathrm{t}_2}(\lambda_1, \mathrm{E}'), \\ P_2 &= e(\varphi_1), \\ P_3 &= e(\lambda_1), \\ P_4 &= \mathrm{Follow}(\varphi_1, \mathrm{E}'). \end{aligned}$$

We have for all $x \in \mathrm{Pos(t)}$:

$$\mathrm{Follow}_t(x, \mathrm{E}') = \mathrm{Follow}_{t_1}(x, \mathrm{E}') + \mathrm{Follow}_{t_2}(x, \mathrm{E}').$$

Case $x \in \mathrm{Pos}(\mathrm{t}_1)$**:**
One has

$$\begin{aligned}
\mathrm{Follow}_{t_2}(x, \mathrm{E}') &\overset{\textit{Relation (15)}}{=} \sum_{\substack{x \preceq \lambda \prec \mathrm{E}' \\ f(\lambda)=\varphi}} \pi(x,\lambda) e_{t_2}(\varphi) \\
&= \sum_{\substack{x \preceq \lambda \prec \mathrm{E}' \\ f(\lambda)=\varphi}} \pi(x,\lambda_1) \otimes \pi(\lambda_1,\lambda) e_{t_2}(\varphi) \\
&= \pi(x,\lambda_1) \sum_{\substack{\lambda_1 \preceq \lambda \prec \mathrm{E}' \\ f(\lambda)=\varphi}} \pi(\lambda_1,\lambda) e_{t_2}(\varphi) \\
&\overset{\textit{Relations (12) and (15)}}{=} \langle e(\lambda_1), x\rangle P_1 \\
&= k_x^{P_1} P_1.
\end{aligned}$$

Thus

$$\mathrm{Follow}_t(x, \mathrm{E}') = \mathrm{Follow}_{t_1}(x, \mathrm{E}') + k_x^{P_1} P_1.$$

It is easy to see that $\mathrm{supp}(\mathrm{Follow}_{t_1}(x, \mathrm{E}')) \cap \mathrm{supp}(P_1) = \emptyset$. If $k_x^{P_1} \neq 0$, $(k_x^{P_1}, P_1)$ constitues the first element of our follow decomposition $\mathrm{dec}(x,t)$ of x in t. We recursively apply the same decomposition process to $\mathrm{Follow}_{t_1}(x, \mathrm{E}')$. Thus we get

$$\mathrm{dec}(x, \mathrm{t}) = \begin{cases} \mathrm{dec}(x, \mathrm{t}_1) & \text{if } k_x^{P_1} = 0, \\ \mathrm{dec}(x, \mathrm{t}_1) \cup \{(k_x^{P_1}, P_1)\} & \text{otherwise.} \end{cases}$$

Case $x \in \mathrm{Pos}(\mathrm{t}_2)$**:**

Lemma 5.1. *Let x be a position in $\mathrm{Pos}(\mathrm{t}_2)$ and let (λ, φ) be a follow link in Δ_x. If $\mathrm{supp}(e(\varphi)) \cap \mathrm{Pos}(\mathrm{t}_1) \neq 0$, then $\varphi_1 \preceq \varphi$.*

Proof. We have $(\lambda, \varphi) \in \Delta_x$, Then $x \preceq \lambda$.
If $\mathrm{supp}(e(\varphi)) \cap \mathrm{Pos}(\mathrm{t}_1) \neq 0$, thus $\varphi \preceq \varphi_1$ or $\varphi_1 \preceq \varphi$. Suppose that $\varphi \preceq \varphi_1$, so one has $x \preceq \lambda \preceq \lambda_1$. Contradiction with $x \in \mathrm{Pos}(\mathrm{t}_2)$. □

One has

$$\begin{aligned}
\mathrm{Follow}_{t_1}(x,\mathrm{E}') &\overset{\textit{Relation (16)}}{=} \sum_{\substack{x\preceq\lambda\prec\mathrm{E}'\\ f(\lambda)=\varphi}} \pi(x,\lambda)e_{t_1}(\varphi)\\
&\overset{\textit{Lemma 5.1}}{=} \sum_{\substack{\varphi_1\preceq\varphi\prec\mathrm{E}'\\ f(\lambda)=\varphi}} \pi(x,\lambda)e(\varphi)\\
&\overset{\textit{Relation (12)}}{=} [\sum_{\substack{\varphi_1\preceq\varphi\prec\mathrm{E}'\\ f(\lambda)=\varphi}} \langle e(\lambda),x\rangle\otimes\pi(\varphi,\varphi_1)]e(\varphi_1)\\
&= \langle \sum_{\substack{\varphi_1\preceq\varphi\prec\mathrm{E}'\\ f(\lambda)=\varphi}} e(\lambda)\pi(\varphi,\varphi_1),x\rangle e(\varphi_1)\\
&= \langle P_4,x\rangle P_2\\
&= k_x^{P_2}P_2.
\end{aligned}$$

Thus

$$\mathrm{Follow}_t(x,\mathrm{E}') = \mathrm{Follow}_{t_2}(x,\mathrm{E}') + k_x^{P_2}P_2.$$

It is easy to see that $\mathrm{supp}(\mathrm{Follow}_{t_2}(x,\mathrm{E}'))\cap\mathrm{supp}(P_2)=\emptyset$. In this case if $k_x^{P_2}\neq 0$, $(k_x^{P_2},P_2)$ constitues the first element of our follow decomposition $\mathrm{dec}(x,\mathrm{t})$ of x in t. We recursively apply the same decomposition process to $\mathrm{Follow}_{t_2}(x,\mathrm{E}')$. Thus we get

$$\mathrm{dec}(x,\mathrm{t}) = \begin{cases} \mathrm{dec}(x,\mathrm{t}_2) & \text{if } k_x^{P_2}=0,\\ \mathrm{dec}(x,\mathrm{t}_2)\cup\{(k_x^{P_2},P_2)\} & \text{otherwise.}\end{cases}$$

If $|\,\mathrm{Pos}(\mathrm{t})|=1$

Using the same idea of the previous cases, we get

$$\mathrm{dec}(x,\mathrm{t}) = \{(\langle P_0,x\rangle,x)\},$$

with $P_0=\sum_{x\preceq\lambda\prec\mathrm{E}'}(\pi(x,\lambda)\otimes\pi(f(\lambda),x))x$.

The following example illustrates the different stages of the recursive procedure.

Example 5.2. Consider the regular $\mathbb{K}$-expression $\mathrm{E}=(((a+\frac{1}{3})+(b+\frac{1}{6}))(b+1))^*$. One has

Step 1.

$|\,\mathrm{Pos}(\mathrm{T}(\mathrm{E}'))|=4$. We decompose the tree $\mathrm{t}=\mathrm{T}(\mathrm{E}')$ into two subtrees t_1 and t_2 such that t_1 be the subtree representing the subexpression $(a_1+\frac{1}{3})+(b_2+\frac{1}{6})$ and $\mathrm{t}_2=\mathrm{t}\setminus\mathrm{t}_1$. In this case one has

$$\begin{aligned}
\mathrm{dec}(a_1,\mathrm{t}) &= \mathrm{dec}(a_1,\mathrm{t}_1)\cup\{(1,(2b_3+2\sharp))\},\\
\mathrm{dec}(b_2,\mathrm{t}) &= \mathrm{dec}(b_2,\mathrm{t}_1)\cup\{(1,(2b_3+2\sharp))\},\\
\mathrm{dec}(b_3,\mathrm{t}) &= \mathrm{dec}(b_3,\mathrm{t}_2)\cup\{(1,(2a_1+2b_2))\}.
\end{aligned}$$

Step 2.
Recursively, we decompose the subtree t_1 into two subtrees t_{1_1} (the subtree representing the subexpression $a_1 + \frac{1}{3}$ and $\mathrm{t}_{1_2} = \mathrm{t}_1 \setminus \mathrm{t}_{1_1}$). Similarly, we divide the subtree t_2 into two subtrees t_{2_1} (the subtree representing the subexpression $b_3 + 1$ and $\mathrm{t}_{2_2} = \mathrm{t}_2 \setminus \mathrm{t}_{2_1}$). Thus we get

$$\begin{aligned}
\mathrm{dec}(a_1, \mathrm{t}) &= \mathrm{dec}(a_1, \mathrm{t}_{1_1}) \cup \{(2, (b_2))\} \cup \{(1, (2b_3 + 2\sharp))\},\\
\mathrm{dec}(b_2, \mathrm{t}) &= \mathrm{dec}(b_2, \mathrm{t}_{1_2}) \cup \{(2, (a_1))\} \cup \{(1, (2b_3 + 2\sharp))\},\\
\mathrm{dec}(b_3, \mathrm{t}) &= \mathrm{dec}(b_3, \mathrm{t}_{2_1}) \cup \{(2, (\sharp))\} \cup \{(1, (2a_1 + 2b_2))\}.
\end{aligned}$$

Step 3.
One has $|\,\mathrm{Pos}(\mathrm{t}_{1_1})| = |\,\mathrm{Pos}(\mathrm{t}_{1_2})| = |\,\mathrm{Pos}(\mathrm{t}_{2_1})| = 1$, thus:

$$\begin{aligned}
\mathrm{dec}(a_1, \mathrm{t}) &= \{(2, (a_1))\} \cup \{(2, (b_2))\} \cup \{(1, (2b_3 + 2\sharp))\},\\
\mathrm{dec}(b_2, \mathrm{t}) &= \{(2, (b_2))\} \cup \{(2, (a_1))\} \cup \{(1, (2b_3 + 2\sharp))\},\\
\mathrm{dec}(b_3, \mathrm{t}) &= \{(2, (b_3))\} \cup \{(2, (\sharp))\} \cup \{(1, (2a_1 + 2b_2))\}.
\end{aligned}$$

Thus the resulting set of *common follow polynomials* $\mathcal{F}_{S(\mathrm{E})}$ associated with $S(\mathrm{E})$ is

$$\mathcal{F}_{S(\mathrm{E})} = \{2a_1 + 2b_2 + b_3 + 2\sharp\} \cup \{(a_1), (b_2), (b_3), (\sharp), (2a_1 + 2b_2), (2b_3 + 2\sharp)\}.$$

Using this procedure, we can prove in similar way as the Boolean case the following Lemma.

Lemma 5.3 (*cf.* [11])**.** *For $\mathcal{F}_{S(\mathrm{E})}$ the following holds:*

(1) $|\mathcal{F}_{S(\mathrm{E})}| = O(|\,\mathrm{E}\,|)$,
(2) $\sum_{P \in \mathcal{F}_{S(\mathrm{E})}} |\,\mathrm{supp}(P)| = O(|\,\mathrm{E}\,| \log |\,\mathrm{E}\,|)$,
(3) $|\,\mathrm{dec}(x)| = O(\log |\,\mathrm{E}\,|)$.

6. Efficient CFP system computation

In this section, we first give a naive cubic implementation (Algorithm 6.1) of the CFP system procedure described in the previous section. This constitues our starting point for the efficient implementation of the CFP system procedure. Next, we show that Algorithm 6.1 runs in quadratic time. After, we show that Algorithm 6.1 contains some redundants computations, that can be eliminated. Finally, we prove that the CFP system procedure can be efficiently implemented in $O(|\,\mathrm{E}\,| \log(|\,\mathrm{E}\,|))$ time.

Algorithm 6.1.
CFPS(t)
Begin
/ Input: a subtree* t *of* T(E′) */
/ Output: return the set of all follow decompositions of positions in* t */
1. *Let* λ *be the root of* tl
2. ***if*** *(*| Pos(t)|=1*)*
3. ***then***
4. *Let* $x \in$ Pos(t)
5. P_0:=*Restriction*(Follow$(x, \mathrm{E}'), \mathrm{t})$
6. dec(x):=$\{(\langle P_0, x\rangle, x)\}$
7. ***else***
8. *Divide* t *into* t_1 *and* $\mathrm{t}_2 = \mathrm{t} \setminus \mathrm{t}_1$ *such that* $\frac{|\,\mathrm{Pos(t)}\,|}{3} \leq |\,\mathrm{Pos}(\mathrm{t}_1)\,| \leq \frac{2|\,\mathrm{Pos(t)}\,|}{3}$
9. *Let* λ_1 *be the root of* tl_1
10. *Let* φ_1 *be the root of* tf_1
14. P_1:=*Restriction*(Follow$(\lambda_1, \mathrm{E}'), \mathrm{t}_2)$
15. P_2:=$e(\varphi_1)$
16. P_3:=$e(\lambda_1)$
17. P_4:=Follow(φ_1, E')
13. dec := $CFPS(t_1) \cup CFPS(t_2)$
18. ***for*** $x \in$ Pos(t_1) ***do***
19 $k_x^{P_1}$:=$\langle P_3, x\rangle$
dec(x):=dec$(x) \cup \{(k_x^{P_1}, P_1)\}$
20. ***for*** $x \in$ Pos(t_2) ***do***
21 $k_x^{P_2}$:=$\langle P_4, x\rangle$
dec(x):=dec$(x) \cup \{(k_x^{P_2}, P_2)\}$
22. ***endif***
23. ***return(***dec***)***
End

Let $T(|\,\mathrm{t}\,|)$ be the time complexity of the procedure $CFPS(\mathrm{t})$ where t is a subtree of T(E′). We have

$$T(|\,\mathrm{t}\,|) \quad = \quad \begin{cases} T(\frac{2|\,\mathrm{t}\,|}{3}) + T(\frac{|\,\mathrm{t}\,|}{3}) + g(|\,\mathrm{t}\,|) & \text{if } |\,\mathrm{Pos}(t)| > 1, \\ g(|\,\mathrm{t}\,|) & \text{otherwise,} \end{cases} \tag{17}$$

where $g(|\,\mathrm{t}\,|)$ is the time complexity of computation of the polynomials P_i for $0 \leq i \leq 4$. A naive implementation (*i.e.* $g(|t|) = O(|\,\mathrm{E}\,|^2)$) yields to a cubic complexity on the size of E. We can therefore reduce this complexity by improving the complexity of P_i computation.

In the following section, we first show how these polynomials can be computed in linear time on the size of E (*i.e.* $g(|\,\mathrm{t}\,|) = O(|\,\mathrm{E}\,|)$), next we present a refinement of Algorithm 6.1 in order to compute these polynomials in linear time on the size of the subtree t (*i.e.* $g(|\,\mathrm{t}\,|) = O(|\,\mathrm{t}\,|)$).

6.1. Efficient follow computation

Let t be a subtree deduced from a decomposition of $\mathrm{T}(\mathrm{E}')$. Let λ_1 be the root of tl, and let λ_2 a node in tl such that $\lambda_1 \preccurlyeq \lambda_2$ and

$$\mathrm{Follow}(\lambda_1, \lambda_2) \quad = \quad \sum_{\lambda_1 \preccurlyeq \lambda \prec \lambda_2} \pi(\lambda_1, \lambda) e(f(\lambda)).$$

The supports of polynomials $(e(f(\lambda)))_{\lambda_1 \preccurlyeq \lambda \prec \lambda_2}$ are not disjoints. So the computation of $\mathrm{Follow}_{\mathrm{t}}(\lambda_1, \lambda_2)$ according to equation (15) requires a quadratic time on the size of t.

Let λ' and λ'' be to nodes such that $\lambda_1 \preccurlyeq \lambda' \prec \lambda'' \preccurlyeq \lambda_2$ and $\mathrm{supp}(e(f(\lambda')) \cap \mathrm{supp}(e(f(\lambda)) \neq \emptyset$.

In this case we have

$$e(f(\lambda'')) \quad = \quad \pi(f(\lambda''), f(\lambda')) e(f(\lambda')) + e(f(\lambda'') - f(\lambda'))$$

where $e(f(\lambda'') - f(\lambda'))$ denotes the polynomial induced by the subtree $\mathrm{t}_{\lambda''} \setminus \mathrm{t}_{\lambda'}$. So the polynomial

$$\pi(\lambda_1, \lambda') e(f(\lambda')) + \pi(\lambda_1, \lambda'') e(f(\lambda''))$$

can be written

$$[\pi(\lambda_1, \lambda') \oplus \pi(\lambda_1, \lambda'') \otimes \pi(f(\lambda''), f(\lambda')\,] e(f(\lambda')) + \pi(\lambda_1, \lambda'') e(f(\lambda'') - f(\lambda')).$$

Algorithm 6.2.

```
Follow(λ1,coef,λ2) : polynomial
// λ1 and λ2 are two nodes s.t. λ1 ≼ λ2
Begin
1.   coef:=π(λ1, λ2) ⊕ coef ⊗π(father(f(λ2)), f(λ2))
2.   case (arity(f(λ2)) of
3.   0 :
4.      If ((coef ≠ 0) and (f(λ2) is a position))
5.      then
6.         return(coef f(λ2))
7.      else
8.         return(0)
9.      endif
10.  1 :
11.     return(Follow(λ1,coef,son(f(λ2))))
12.  2 :
13.     return(Follow(λ1,coef,leftson(f(λ2)))+Follow(λ1,coef,rightson(f(λ2))))
14.  endcase
End
```

Here, supports of polynomials $e(f(\lambda'))$ and $e(f(\lambda'') - f(\lambda'))$ are disjoints. Algorithm 6.2 is based on this formula. The call Follow(λ_2,1,λ_2), computes the

polynomial $\text{Follow}(\lambda_1, \lambda_2)$ in linear time on the size of t. Indeed, each node in tl_2 is treated only once.

As polynomials Follow are ordered on positions, the restriction of $\text{Follow}(\lambda_1, \lambda_2)$, $\mathit{Restriction}(\text{Follow}(\lambda_1, \lambda_2), \text{t}')$, to some subtree t' of t needs a linear time on the size of t.

In a similar way, the polynomial $\text{Follow}(\varphi_1, \varphi)$ is computed in linear time w.r.t. the size of t where φ is the root of tf and φ_1 is a node in tf.

Lemma 6.3. *Let* E *be a regular* $\mathbb{K}$*-expression. Let* t,t' *be subtrees of* $\text{T}(\text{E}')$*. Let* λ_1 *be the root of* tl*, and let* λ_2 *a node in* tl *such that* $\lambda_1 \preccurlyeq \lambda_2$*. Then the polynomial* $\text{Follow}_{\text{t}'}(\lambda_1, \lambda_2)$ *can be computed in* $O(\max(|\,\text{t}\,|, |\,\text{t}'\,|)$.

Now polynomials P_i, for $0 \leq i \leq 4$, are computed in linear time on the size of E (*i.e.* $(g(|\,\text{t}\,|) = O(|\,\text{E}\,|))$. So Algorithm 6.1 runs in quadratic time on the size of E.

In the following section we show that Algorithm 6.1 contains some redundant computations that can be avoided. This improvement yields to an $O(|\,\text{t}\,|\log(|\,\text{t}\,|))$ time complexity.

6.2. Redundant computations

At each call of CFPS procedure we compute the following polynomials Follow (λ_1, E'), $\text{Follow}(\varphi_1, \text{E}')$. We have

$$\begin{aligned} \text{Follow}(\lambda_1, \text{E}') &= \text{Follow}(\lambda_1, \lambda) + \pi(\lambda_1, \lambda)\,\text{Follow}(\lambda, \text{E}') \\ \text{Follow}(\varphi_1, \text{E}') &= \text{Follow}(\varphi_1, \varphi) + \pi(\varphi_1, \varphi)\,\text{Follow}(\varphi_1, \text{E}') \end{aligned}$$

where λ and φ are respectively the root of tl and tf and λ_1 and φ_1 are respectively the root of tl_1 and tf_1. Denote by $R(\text{t})$ the polynomial $\text{Follow}(\lambda, \text{E}')$. Then, we get

$$R(\text{t}_1) = \text{Follow}(\lambda_1, \lambda) + \pi(\lambda_1, \lambda)R(\text{t}),$$

and as the root of t is the same as t_2 we get

$$R(\text{t}_2) = R(\text{t}).$$

In a similar way $\text{Follow}(\varphi_1, \text{E}')$ is computed. Denote by $S(\text{t})$ the polynomial $\text{Follow}(\varphi, \text{E}')$. So we get

$$S(\text{t}_1) = \text{Follow}(\varphi_1, \varphi) + \pi(\varphi_1, \varphi)S(\text{t})$$

and

$$S(\text{t}_2) = S(\text{t}).$$

It is obvious that if $R(\mathrm{t})$ and $S(\mathrm{t})$ are computed then $\mathrm{Follow}_{\mathrm{t}_2}(\lambda_2, \mathrm{E}')$ and $\mathrm{Follow}_{\mathrm{t}_2}(\varphi_1, E')$ can be computed in $O(|\,\mathrm{t}\,|)$ time. Finlay Algorithm 6.1 runs in $O(|\,\mathrm{t}\,|\log(|\,\mathrm{t}\,|))$ time.

Algorithm 6.4.
CFPS(t, R, S)
Begin
/ Input: a subtree* t *of* $\mathrm{T}(\mathrm{E}')$ */
/ Output: return the set of all follow decompositions of positions in* t */
1. *Let* λ *be the root of* tl
2. ***if*** $(|\,\mathrm{Pos(t)}\,|{=}1)$
3. ***then***
4. *Let* $x \in \mathrm{Pos(t)}$
5. P_0*:=Restriction*$(\mathrm{Follow}(x,\lambda) + \pi(x,\lambda)R, \mathrm{t})$
6. $\mathrm{dec}(x)$:=$\{\langle P_0, x\rangle, x\}$
7. ***else***
8. *Divide* t *into* t_1 *and* $\mathrm{t}_2 = \mathrm{t} \setminus \mathrm{t}_1$ *such that* $\dfrac{|\,\mathrm{Pos(t)}\,|}{3} \leq |\,\mathrm{Pos}(\mathrm{t}_1)\,| \leq \dfrac{2|\,\mathrm{Pos(t)}\,|}{3}$
9. *Let* λ_1 *be the root of* tl_1
10. *Let* φ_1 *be the root of* tf_1
14. P_1*:=Restriction*$(\mathrm{Follow}(\lambda_1,\lambda) + \pi(\lambda_1,\lambda)R, \mathrm{t}_2)$
15. P_2*:=*$e(\varphi_1)$
16. P_3*:=*$e(\lambda_1)$
17. P_4*:=*$\mathrm{Follow}(\varphi_1,\varphi) + \pi(\varphi_1,\varphi)S$
13. $\mathrm{dec} := CFPS(t_1, \mathrm{Follow}(\lambda_1,\lambda) + \pi(\lambda_1,\lambda)R, \mathrm{Follow}(\varphi_1,\varphi) + \pi(\varphi_1,\varphi)S) \cup CFPS(t_2, R, S)$
18. ***for*** $x \in \mathrm{Pos}(\mathrm{t}_1)$ ***do***
19 $k_x^{P_1}$*:=*$\langle P_3, x\rangle$
$\mathrm{dec}(x)$:=$\mathrm{dec}(x) \cup \{(k_x^{P_1}, P_1)\}$
20. ***for*** $x \in \mathrm{Pos}(\mathrm{t}_2)$ ***do***
21 $k_x^{P_2}$*:=*$\langle P_4, x\rangle$
$\mathrm{dec}(x)$:=$\mathrm{dec}(x) \cup \{(k_x^{P_2}, P_2)\}$
22. ***endif***
23. ***return(***dec***)***
End

Lemma 6.5. *Let* E *be a regular* $\mathbb{K}$*-expression. The CFP system associated to* E *can be computed in* $O(|\,\mathrm{E}\,|\log(|\,\mathrm{E}\,|))$ *time.*

7. CFP automaton computation

In the previous section, we have given an algorithm that compute the CFP system. So the set of states of the CFP automaton is computed. It remains to construct the set of transitions.

Algorithm 7.2 computes in a first stage the set of transitions over $A_{\overline{\mathrm{E}}}$, and in the second stage deduces the set of transitions over the alphabet A_{E}.

Let us now describe this algorithm. Let $P = \beta_1 x_1 + \beta_2 x_2 + \ldots + \beta_l x_l$ be a state in $\mathcal{F}_{S(\mathrm{E})}$. Let $\beta_i x_i$ be a monomial in P. From the definition of $\mathcal{A}_{S(\mathrm{E})}$,

if $\mathrm{dec}(x_i, \mathrm{E}') = \{(k_1, P_1), (k_2, P_2), \cdots, (k_m, P_m)\}$, we create a transition from the state P to each state P_j, $1 \leqslant j \leqslant m$, labeled $\beta_i k_j x_i$.

Now transitions are labeled by letters in $A_{\overline{\mathrm{E}}}$. So in the second stage of the algorithm, we replace each symbol $x_i \in A_{\overline{\mathrm{E}}}$ by $h(x_i)$. Next for all transitions going from a state P to a state P' labeled by x_i such that $h(x_i) = x$, we merge these transitions into a unique one (P, x, P') with $\gamma(P, x, P') = \bigoplus_{\substack{h(x_i)=x \\ x_i \in A_{\overline{\mathrm{E}}}}} \gamma(P, x_i, P')$.

Finally, as $\sum_{P \in \mathcal{F}_{S(\mathrm{E})}} |\,\mathrm{supp}(P)\,| = O(|\,\mathrm{E}\,|\log(|\,\mathrm{E}\,|))$ and $|\,\mathrm{dec}(x, \mathrm{E}')\,| = O(\log(|\,\mathrm{E}\,|))$, lines $1-7$ and lines $9-16$ of procedure Transitions() are achieved in $O(|\,\mathrm{E}\,|\log^2(|\,\mathrm{E}\,|))$ times.

Theorem 7.1. *Let* E *be a regular* $\mathbb{K}$*-expression. The CFP automaton* $\mathcal{A}_{S(\mathrm{E})}$ *can be computed in* $O(|\,\mathrm{E}\,|\log^2(|\,\mathrm{E}\,|))$ *time.*

Algorithm 7.2.
Transitions()
Begin
1. *// Transitions over* $A_{\overline{\mathrm{E}}}$
1. ***for*** $P \in \mathcal{F}_{S(\overline{\mathrm{E}})}$ ***do***
2. *//* $P = \beta_1 x_1 + \ldots + \beta_l x_l$
3. ***for*** $x \in \mathrm{supp}(P)$ ***do***
4. *//* $\mathrm{supp}(P) = \{x_1, x_2, \ldots, x_l\}$
5. ***for*** $(k, P') \in \mathrm{dec}(x, \mathrm{E}')$ ***do***
6. *//* $\mathrm{dec}(x, \mathrm{E}') = \{(k_1, P_1), (k_2, P_2), \ldots, (k_m, P_m)\}$
7. *Create a transition labeled* (P, x, P')
and set $\gamma(P, x, P') = \langle P, x\rangle \otimes k$
8. *// Transitions over* A_{E}
9. ***for*** $P \in \mathcal{F}_{S(\mathrm{E})}$ ***do***
10. *Let* $q^+(P) = \{(x_1, P_1) \mid (P, x_i, P_i) \in \delta_{\mathcal{A}_{S(\overline{\mathrm{E}})}}\}$
12. ***for*** $a \in A_{\mathrm{E}}$ ***do***
13. ***for*** $(x, P') \in q^+(P)$ ***do***
14. $\gamma(P, a, P')$=0
15. ***for*** $\alpha(x, P') \in q^+(P)$ ***do***
16. $\gamma(P, h(x), P')$=$\gamma(P, h(x), P') \oplus \gamma(P, x, P')$
End

8. Conclusion

We have shown that Hromkovič *et al.* construction can be generalized to weighted regular expressions and presented an efficient computation of CFP automaton.

References

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

[2] A. Brüggemann-Klein, Regular expressions into finite automata. *Theoret. Comput. Sci.* **120** (1993) 197–213.

[3] J. Berstel and C. Reutenauer. *Rational series and their languages*. Springer-Verlag, Berlin (1988).

[4] P. Caron and M. Flouret, Glushkov construction for series: the non commutative case. *Int. J. Comput. Math.* **80** (2003) 457–472.

[5] P. Caron and D. Ziadi, Characterization of Glushkov automata. *Theoret. Comput. Sci.* **231** (2000) 75–90.

[6] J.-M. Champarnaud and D. Ziadi, *Computing the equation automaton of regular expression in* $O(s^2)$ *space and time*, in CPM 2001, Combinatorial Pattern Matching, edited by A. Amir and G.M. Landau. *Lect. Notes Comput. Sci.* **2089** (2001) 157–168.

[7] J.-M. Champarnaud, E. Laugerotte, F. Ouardi and D. Ziadi, From regular weighted expressions to finite automata. *Int. J. Fond. Comput. Sci.* **15** (2004) 687–700.

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

[9] C. Hagenah and A. Muscholl, Computing ε-free NFA from regular expression in $O(n \log^2(n))$ time. *RAIRO-Theor. Inf. Appl.* **34** (2000) 257–277.

[10] U. Hebisch and H.J. Weinert, *Semirings: algebraic theory and applications in computer science*. World Scientific, Singapore (1993).

[11] U. Hromkovič, J. Seibert and T. Wilke, Translating regular expressions into small ε-free nondeterministic finite automata. *J. Comput. System Sci.* **62** (2001) 565–588.

[12] L. Ilie and S. Yu, *Algorithms for computing small NFAs*, in *Proc 27th MFCS*, Warszawa, 2002, edited by K. Diks and W. Rytter. *Lect. Notes Comput. Sci.* (2002) 328–340.

[13] W. Kuich and J. Salomaa, *Semirings, automata, languages*. Springer-Verlag, Berlin (1986).

[14] S. Lombardy and J. Sakarovitch, Derivatives of regular expression with multiplicity, Proc. of MFCS 2002. *Lect. Notes Comput. Sci.* **2420** (2002) 471–48.

[15] R.F. McNaughton and H. Yamada, Regular expressions and state graphs for automata. *IEEE T. Electron. Comput.* **9** (1960) 39–47.

[16] M.P. Schützenberger, On the definition of a family of automata. *Inform. Control* **6** (1961) 245–270.

[17] D. Ziadi, Algorithmique parallèle et séquentielle des automates. *Thesis*, LIR report, Université de Rouen (1996).

[18] D. Ziadi, Quelques aspects théoriques et algorithmiques des automates. *Thesis*, Université de Rouen (2002).

[19] D. Ziadi, J.-L. Ponty and J.-M. Champarnaud, Passage d'une expression rationnelle à un automate fini non-déterministe. *Bull. Belg. Math. Soc. Simon Stevin* **4** (1997) 177–203.

Communicated by C. Choffrut.
Received March 23, 2005. Accepted June 22, 2007.