

Subset construction complexity* for homogeneous automata, position automata and ZPC-structures

J.-M. Champarnaud
Université de Rouen, LIFAR
76821 Mont-Saint-Aignan Cedex, France

Abstract

The aim of this paper is to investigate how subset construction performs on specific families of automata. A new upper bound on the number of states of the subset-automaton is established in the case of homogeneous automata. The complexity of the two basic steps of subset construction, i.e. the computation of deterministic transitions and the set equality tests, is examined depending on whether the nondeterministic automaton is an unrestricted one, an homogeneous one, a position one or a ZPC-structure, which is an implicit construction for a position automaton.

Keywords: Subset construction; Homogeneous automaton; Position sets; Position automaton; ZPC-structure.

1 Introduction

Automata determinization may be exponential, whereas most of automata operations are polynomial. It is not possible to avoid this behaviour in the general case [16]. It is nevertheless important to carefully handle the implementation of the determinization algorithm when designing automata software tools, in order to preserve as well as possible the performances of the whole system.

Two basic computation steps are carried on all along the determinization process: the computation of a deterministic transition, which yields a subset of the set of nondeterministic states, and set equality testing which makes it possible to decide whether a deterministic transition generates a new state or not.

Concerning the computation of the set of deterministic transitions, the choice of the data structure implementing the set of nondeterministic transitions has much influence on the complexity. Softwares such as AUTOMATE [3], INR [6] and GRAIL [17] make use of a representation in which both transitions with

*This work is a contribution to the Automate software development project carried on by A.I.A. (Algorithmics and Implementation of Automata) Working Group, L.I.F.A.R. Contact: {Champarnaud, Ziadi}@dir.univ-rouen.fr.

the same origin and transitions with the same origin and the same symbol are contiguous. Johnson and Wood [7] have studied the efficiency of various sorting procedures applied to subsets computed from such data structures.

On the other hand, the choice of the data structure encoding subsets has a significant influence on the complexity of set equality testing. The number of integer comparisons involved by the overall set equality tests is $O(\sqrt{n}2^{2n})$ when using lists and $O(n^2 \log(n)2^n)$ when using balanced search trees [15]. There exist investigations that are motivated by the will to improve such complexities, for instance Leslie *et al.* [9, 10].

The aim of this paper is to investigate how subset construction performs on specific automata, in particular on position automata. The position automaton of a regular expression is defined by the algorithm described in [4] and in [11]. The author and his co-researchers [20, 13] have designed a linear space and time representation of the position automaton of a regular expression, which is based on two state forests connected by a set of links. The so-called ZPC-structure leads to an output sensitive implementation of the conversion of a regular expression into an automaton and to an efficient algorithm for testing the membership of a word to a regular language [15].

The impact that the specific properties of position automata have on subset construction are analyzed. Firstly, a position automaton is homogeneous and we establish a new upper bound on the number of states of the automaton resulting from the determinization of an homogeneous automaton; this bound is much smaller in many cases than the standard $2^n - 1$ bound which holds for unrestricted automata. We also give an estimation of the complexity of the transition computation and of the set equality testing in the case of an homogeneous automaton. Secondly, a position automaton can be implemented as a ZPC-structure, whose main feature is to allow the computation of disjoint unions of transition sets thanks to a specific encoding of position sets. We show how to make use of this representation in order to reduce the complexity of the two basic steps of the determinization process.

Basic definitions and useful notations are gathered together in the next section. Section 3 recalls how subset construction performs on unrestricted automata, and analyzes its complexity. Section 4 deals with homogeneous automata, section 5 with position automata and section 6 with ZPC-structures. In each of these sections, algorithmic refinements of subset construction are provided, as well as new complexity bounds.

2 Definitions and notations

A reminder of the definitions and notations which are useful for the description of our results will be found in this section. For further details about regular languages and finite automata, the references [5, 18, 12, 19] should be consulted.

A finite automaton is a 5-tuple $\mathcal{M} = (Q, \Sigma, \delta, I, F)$ where Q is a (finite) set of states, Σ is a (finite) alphabet, $I \subseteq Q$ is the set of initial states, $F \subseteq Q$ is the set of final states, and δ is the transition function. The automaton \mathcal{M}

is *deterministic* (\mathcal{M} is a DFA) if and only if $|I| = 1$ and δ is a mapping from $Q \times \Sigma$ to Q . Otherwise \mathcal{M} is a *NFA* and δ is a mapping from $Q \times \Sigma$ to 2^Q . The automaton \mathcal{M} is *complete* if and only if δ is a full mapping.

The following overloadings of the transition function will be used:

$$\delta(p) = \bigcup_{a \in \Sigma} \delta(p, a), \quad \forall p \in Q \quad (1)$$

$$\delta(P, a) = \bigcup_{p \in P} \delta(p, a), \quad \forall P \subseteq Q, \forall a \in \Sigma \quad (2)$$

$$\delta(P) = \bigcup_{a \in \Sigma} \delta(P, a), \quad \forall P \subseteq Q \quad (3)$$

A *path* of \mathcal{M} is a sequence (q_i, a_i, q_{i+1}) , $i = 1, \dots, n$, of consecutive edges. Its *label* is the word $w = a_1 a_2 \dots a_n$. A word $w = a_1 a_2 \dots a_n$ is *recognized* by the automaton \mathcal{M} if there is a path with label w such that $q_1 \in I$ and $q_{n+1} \in F$. The language $L(\mathcal{M})$ *recognized* by the automaton \mathcal{M} is the set of words which it recognizes. Two automata \mathcal{M} and \mathcal{M}' are *equivalent* if and only if they recognize the same language. A state is *accessible* if and only if there is a path from an initial state to this state.

A regular expression over an alphabet Σ is generated by recursively applying operators ‘+’ (union), ‘.’ (concatenation) and ‘*’ (Kleene star) to atomic expressions (i.e. every symbol of Σ , the empty word and the empty set). A language is regular if and only if it can be denoted by a regular expression. The *length* of a regular expression E , denoted by $|E|$, is the number of occurrences of operators and symbols in E . The *alphabetic width* of E , denoted by $\|E\|$, is the number of occurrences of symbols in E . Notice that these two parameters are linearly related as far as sequences of star operators and occurrences of the empty word and of the empty set are carefully preprocessed.

Kleene’s theorem [8] states that a language is regular if and only if it is recognized by a finite automaton. Computing the position automaton of a regular expression [4, 11] is a constructive proof of the direct part of this theorem.

3 Subset construction

Subset construction takes an NFA \mathcal{M} as input, and yields an equivalent DFA denoted by $\mathcal{D}_{\mathcal{M}}$ as output. The automaton $\mathcal{D}_{\mathcal{M}}$ is the *subset-automaton* of \mathcal{M} .

3.1 The subset-automaton

Definition 1 Let $\mathcal{M} = (Q, \Sigma, \delta, I, F)$ be an NFA. The subset-automaton of \mathcal{M} is the automaton $\mathcal{D}_{\mathcal{M}} = (Q', \Sigma, \delta', \{0'\}, F')$ defined as follows [5, 19]:

- *Set of states:* A deterministic state is a set of nondeterministic states; for all q' in Q' , we have $q' \subseteq Q$.

- *Initial state:* The initial state in $\mathcal{D}_{\mathcal{M}}$, denoted by $0'$, is the set I of initial states in \mathcal{M} .
- *Set of transitions:* Let q' be a deterministic state and let a be a symbol in Σ . If the transition from q' on symbol a is defined, then, by construction, its target is the state $\delta'(q', a)$ such that:

$$\delta'(q', a) = \bigcup_{q \in q'} \delta(q, a) \quad (4)$$

- *Set of final states:* A deterministic state is final if and only if it contains at least one final nondeterministic state: $q' \in F' \Leftrightarrow q' \cap F \neq \emptyset$.

Lemma 1 *Let q' be a deterministic state. The transition function δ' deduces from δ according to the following formulas:*

$$\delta'(q', a) = \delta(q', a), \forall a \in \Sigma \quad (5)$$

$$\delta'(q') = \delta(q') \quad (6)$$

Proof. By Formula (4), we have: $\delta'(q', a) = \bigcup_{q \in q'} \delta(q, a)$. Since $q' \subseteq Q$, by Formula (2), we get: $\bigcup_{q \in q'} \delta(q, a) = \delta(q', a)$. Consequently, we have: $\delta'(q', a) = \delta(q', a)$. From Formula (1) we deduce: $\delta'(q') = \bigcup_{a \in \Sigma} \delta'(q', a)$. By Formula (3) we have: $\delta(q') = \bigcup_{a \in \Sigma} \delta(q', a)$. Thus we get: $\delta'(q') = \delta(q')$. ■

3.2 The standard bound on the number of states of the subset-automaton

In the following, the automata \mathcal{M} and $\mathcal{D}_{\mathcal{M}}$ are not necessarily complete. Let n (resp. n') be the number of states in \mathcal{M} (resp. in $\mathcal{D}_{\mathcal{M}}$). The upper bound $n' \leq 2^n - 1$ obviously holds. As far as unrestricted automata are concerned, it is a least upper bound, as established by Rabin and Scott [16].

There exist two possible implementations of subset construction:

1. The first one computes the transitions of the $2^n - 1$ potential states and then makes the DFA trim.
2. The second one computes the transitions of accessible states only. The main point is to decide whether a target state is a new state. It induces many time-consuming identity tests involving a potentially new subset and already produced subsets.

From now on, we shall investigate the construction based on accessibility and we shall compare how it performs on unrestricted automata and on specific families of automata.

3.3 The standard complexity

Complexity results about unrestricted automata are summarized in the following theorem:

Theorem 1 *Let \mathcal{M} be an unrestricted automaton on the alphabet Σ , and $\mathcal{D}_{\mathcal{M}}$ be the subset-automaton of \mathcal{M} . Let n (resp. n') be the number of states in \mathcal{M} (resp. in $\mathcal{D}_{\mathcal{M}}$). The automaton $\mathcal{D}_{\mathcal{M}}$ can be computed with the following complexity:*

1. *Upper bound on the number of states: $n' \leq 2^n - 1$.*
2. *Transition computation:*
 - (a) *$O(n^2)$ for one transition,*
 - (b) *$O(|\Sigma|n^2)$ for the set of transitions of one state,*
 - (c) *$O(|\Sigma|n^2n')$ for the set of transitions of the subset-automaton.*
3. *Set equality testing:*
 - (a) *$O(\sqrt{n}2^{2n})$ when using lists,*
 - (b) *$O(n^2 \log(n)2^n)$ when using balanced search trees.*

Proof. Let us first examine the complexity of transition computation. By construction, for all q' in Q' and all a in Σ , the target state of the transition from q' on the symbol a is the state $\delta'(q', a) = \bigcup_{q \in q'} \delta(q, a)$. Since $|q'| \leq n$, and since $|\delta(q, a)| \leq n$ for all q in q' , the subset $\delta(q', a)$ is computed with an $O(n^2)$ time complexity.

The complexity of set equality testing depends on the data structure encoding subsets. The results reported in Theorem 1 concern lists and balanced search trees, improved by a hashing w.r.t. subset size. A complete proof can be found in [14].

■

4 Homogeneous automata

This section aims at studying the impact of the homogeneity property on subset construction. We first establish a new upper bound on the number of states of the subset-automaton of an homogeneous automaton. We also give an estimation of the complexity of transition computation and of set equality testing in the homogeneous case.

4.1 The homogeneity property

Definition 2 An automaton $\mathcal{M} = (Q, \Sigma, \delta, I, F)$ is homogeneous if and only if, for all state q in Q , all transitions in q are labeled by the same symbol:

$$\forall r, s \in Q, \forall a, b \in \Sigma, a \neq b \Rightarrow \delta(r, a) \cap \delta(s, b) = \emptyset$$

A state q is said to be *labeled* by a symbol a if and only if all transitions in q are labeled by a . Notice that an initial state may have no in-transition. Assume symbol 0 is not in Σ and let 0 be the label of initial states with no in-transition. Let $Q_a, a \in \Sigma \cup \{0\}$, be the set of states labeled by the symbol a . We have: $Q_0 = I \setminus \bigcup_{a \in \Sigma} Q_a$. The sets Q_a , for all a in $\Sigma \cup \{0\}$, are obviously pairwise disjoint. Assuming \mathcal{M} is accessible, we have the following property:

Lemma 2 The set of states of an homogeneous automaton is partitionned as follows:

$$Q = \bigsqcup_{a \in \Sigma \cup \{0\}} Q_a, \text{ with } Q_0 = I \setminus \bigcup_{a \in \Sigma} Q_a$$

A subset P of Q is said to be *homogeneous* if and only if there exists a symbol a in Σ such that $P \subseteq Q_a$. In this case, we say that P is *labeled* by a .

Lemma 3 Let $\mathcal{M} = (Q, \Sigma, \delta, I, F)$ be an homogeneous automaton. Let Q_a , for all a in Σ , be the set of states labeled by the symbol a . The following properties hold:

1. For all subset P of Q , and for all a in Σ , we have: $\delta(P, a) \subseteq Q_a$.
2. The subset-automaton of an homogeneous automaton \mathcal{M} is homogeneous.
3. For all non-initial state q' in $\mathcal{D}_{\mathcal{M}}$, there exists a symbol a such that $q' \subseteq Q_a$.

Proof.

(1) Let P be a subset of Q . We have: $\delta(P, a) = \bigcup_{q \in P} \delta(q, a)$. Since \mathcal{M} is homogeneous, we have: $\delta(q, a) \subseteq Q_a, \forall q \in Q$. Thus we get: $\delta(P, a) \subseteq Q_a$.

(2) Assume there exist two states x' and y' in Q' and two distinct symbols a and b in Σ , such that $\delta'(x', a) = \delta'(y', b)$. By Lemma 1, it implies that $\delta(x', a) = \delta(y', b)$. Since \mathcal{M} is homogeneous, by (1) we have: $\delta(x', a) \subseteq Q_a$ and $\delta(y', b) \subseteq Q_b$. Since $Q_a \cap Q_b = \emptyset$, we get a contradiction.

(3) Let q' be a non-initial state in $\mathcal{D}_{\mathcal{M}}$. By construction, there exists at least one pair $(r', a) \in Q' \times \Sigma$ such that $\delta'(r', a) = q'$. Notice that since $\mathcal{D}_{\mathcal{M}}$ is homogeneous the symbol a is unique. By construction, $q' = \bigcup_{q \in r'} \delta(q, a)$. Since \mathcal{M} is homogeneous, from (1) we deduce that $q' \subseteq Q_a$. ■

Remark 1 Notice that the subset I , which is the initial deterministic state, is not necessarily homogeneous. If it is, then there exists a symbol a such that $I \subseteq Q_a$. In this case, there may exist a deterministic state r' such that $\delta(r', a) = I$, which implies $0' \in Q'_a$. Otherwise $Q'_0 = \{0'\}$.

4.2 A new upper bound on the number of states in $\mathcal{D}_{\mathcal{M}}$

Lemma 3.3 implies that the number of subsets produced by the determinization of an homogeneous automaton is equal to the overall number of the non-empty subsets of sets Q_a , a in Σ . Since subset I is not necessarily homogeneous, one is added to get at the following upper bound on the number of states in the subset-automaton.

Proposition 1 *Let $\mathcal{M} = (Q, \Sigma, \delta, I, F)$ be an homogeneous automaton and Q_a be the set of states labeled by the symbol a . Then the number n' of states in the subset-automaton of \mathcal{M} is such that:*

$$n' \leq \left(\sum_{a \in \Sigma} 2^{|Q_a|} \right) - |\Sigma| + 1, \text{ with } \sum_{a \in \Sigma} |Q_a| = |Q| - |Q_0|$$

Remark 2 *This new upper bound is generally much smaller than the standard bound. For instance, if there are at least two symbols, and if subsets Q_a have the same size, we get: $n' \leq |\Sigma| \times 2^{|Q|/|\Sigma|} \ll 2^{|Q|} - 1$. The gap increases with the alphabet size, which can be very large in linguistic applications. Let us notice that such a bound could help to decide whether an exhaustive implementation is realistic or not.*

4.3 Transition computation

Let q' be a deterministic state and a be a symbol. We have to compute the subset $\delta'(q', a)$, according to the formula: $\delta'(q', a) = \bigcup_{q \in q'} \delta(q, a)$.

Since \mathcal{M} is homogeneous, for all q in q' , we have: $\delta(q, a) \subseteq Q_a$, and thus $|\delta(q, a)| \leq |Q_a|$. If q' is not an initial state, by Lemma 3.3, there exists a symbol ℓ such that $q' \subseteq Q_\ell$ and thus $|q'| \leq |Q_\ell|$. Consequently, the transition $\delta'(q', a)$ is computed with an $O(|Q_\ell| \times |Q_a|)$ time complexity. On the other hand, the transition $\delta'(0', a)$ is computed with an $O(|I| \times |Q_a|)$ time complexity. Hence the following proposition:

Proposition 2 *For an homogeneous automaton, such that:*

$$Q = \biguplus_{\ell \in \Sigma \cup \{0\}} Q_\ell, \text{ with } Q_0 = I \setminus \bigcup_{\ell \in \Sigma} Q_\ell,$$

the time complexity of the transition computation step is the following:

1. $O(|Q_\ell| \times |Q_a|)$ for the transition $\delta'(q', a)$ from a non-initial state q' labeled with symbol ℓ , and $O(|I| \times |Q_a|)$ for the transition $\delta'(0', a)$, for all a in Σ ,
2. $O(|Q_\ell| \times |Q|)$ for the set of transitions $\delta'(q')$ from a non-initial state q' labeled with symbol ℓ , and $O(|I| \times |Q|)$ for the transition $\delta'(0')$,
3. $O(|Q| \times \sum_{\ell \in \Sigma} |Q_\ell| 2^{|Q_\ell|})$ for the set of transitions of the subset-automaton.

Remark 3 This complexity generally improves the standard one by a factor greater than $|\Sigma|$. For instance, if there are at least two symbols, and if subsets Q_a have the same size, we get: $n' \leq |Q| \times 2^{|\Sigma|/|Q|} \ll |\Sigma| \times |Q|^2 \times 2^{|\Sigma|}$. Let us notice that this improvement is due to homogeneity property; it does not depend on the implementation, which can be based on the raw transition table of the automaton.

Lemma 4 In the homogeneous case, the family of subsets $(\delta'(q', a))_{a \in \Sigma}$, can be deduced from the set $\delta(q')$, according to the formula: $\delta'(q', a) = \delta(q') \cap Q_a$.

Proof. By Lemma 3.3, we have: $\delta(q', a) \subseteq Q_a$, for all a in Σ . Since sets Q_a are pairwise disjoint, we get: $\delta(q') = \bigsqcup_{a \in \Sigma} \delta(q', a)$. By Lemma 1 it implies that $\delta'(q') = \bigsqcup_{a \in \Sigma} \delta'(q', a)$. Finally, we have: $\delta'(q', a) = \delta(q') \cap Q_a$. ■

As a consequence, subsets $(\delta'(q', a))_{a \in \Sigma}$ can be retrieved from $\delta(q')$ set with an overall $O(|Q|)$ time complexity. This property is particularly interesting when the set $\delta(q')$ can be computed with an $O(|Q|)$ time complexity. It is not the case for an arbitrary homogeneous automaton since $\delta(q')$, for a subset q' labeled by ℓ , is computed in $O(|Q_\ell| \times |Q|)$ time. We shall see it is the case for the ZPC-structure representation of a position automaton.

4.4 Set equality testing

Homogeneous bounds reported in the following theorem are deduced from standard bounds, by hashing the data structure implementing subsets w.r.t. the partition of Q into homogeneous subsets.

4.5 Homogeneous complexity

Complexity results about homogeneous automata are summarized in the following theorem:

Theorem 2 Let \mathcal{M} be an homogeneous automaton on the alphabet Σ , such that:

$$Q = \bigsqcup_{l \in \Sigma \cup \{0\}} Q_l, \text{ with } Q_0 = I \setminus \bigcup_{l \in \Sigma} Q_l.$$

Let $\mathcal{D}_\mathcal{M}$ be the subset-automaton of \mathcal{M} . Let n (resp. n') be the number of states in \mathcal{M} (resp. in $\mathcal{D}_\mathcal{M}$). The automaton $\mathcal{D}_\mathcal{M}$ can be computed with the following complexity:

1. Upper bound on the number of states:

$$n' \leq \left(\sum_{a \in \Sigma} 2^{|Q_a|} \right) - |\Sigma| + 1, \text{ with } \sum_{a \in \Sigma} |Q_a| = |Q| - |Q_0|$$

2. Transition computation:

- (a) $O(|Q_\ell| \times |Q_a|)$ for the transition $\delta'(q', a)$ from a non-initial state q' labeled with symbol ℓ , for all a in Σ ,
- (b) $O(|Q_\ell| \times |Q|)$ for the set of transitions $\delta'(q')$ from a non-initial state q' labeled with symbol ℓ ,
- (c) $O(|Q| \times \sum_{\ell \in \Sigma} |Q_\ell| 2^{|Q_\ell|})$ for the set of transitions of the subset-automaton.

3. Set equality testing:

- (a) $O(\sum_{a \in \Sigma} \sqrt{|Q_a|} \times 2^{2 \times |Q_a|})$ when using lists,
- (b) $O(\sum_{a \in \Sigma} |Q_a|^2 \times \log(|Q_a|) \times 2^{|Q_a|})$ when using balanced search trees.

5 Position automata

We first recall the definition of the position automaton of a regular expression. Position automata turn to be homogeneous, which leads to a new bound on the number of deterministic states w.r.t. the number of occurrences of symbols in the expression.

5.1 The position automaton of a regular expression

A regular expression over an alphabet Σ is *linear* if and only if all its symbols are distinct. Let E be a linear expression over Σ . The following sets of symbols are associated to E :

- $Null(E) = \{\varepsilon\}$ if $\varepsilon \in L(E)$ and \emptyset otherwise,
- $First(E)$, the set of symbols that match the first symbol of some word in $L(E)$,
- $Last(E)$, the set of symbols that match the last symbol of some word in $L(E)$,
- $Follow(E, x)$, for all x in Σ : the set of symbols that follow the symbol x in some word of $L(E)$.

Now, let E be a regular expression over Σ . If x is the j^{th} occurrence of a symbol in E , the pair (x, j) , or x_j for short, is the *position* associated to x . The set of positions of E is denoted by $Pos(E)$. Let \bar{E} be the linear expression deduced from E by substituting each symbol by its position. We denote by h the application from $Pos(E)$ to Σ which maps a position to the symbol it is associated to. Let $E = a \cdot (a + b) + (a + b) \cdot (1 + b)$. We have: $Pos(E) = \{a_1, a_2, b_3, a_4, b_5, b_6\}$, $\bar{E} = a_1 \cdot (a_2 + b_3) + (a_4 + b_5) \cdot (1 + b_6)$, $h(a_1) = h(a_2) = h(a_4) = a$ and $h(b_3) = h(b_5) = h(b_6) = b$.

The sets of positions associated to E are straightforwardly deduced from the sets of symbols associated to \bar{E} :

- $Null(E) = Null(\overline{E})$,
- $First(E) = First(\overline{E})$,
- $Last(E) = Last(\overline{E})$,
- $Follow(E, x) = Follow(\overline{E}, x)$, for all x in $Pos(E)$.

Positions of E , added with an initial state, are the states of the position automaton of E , denoted by \mathcal{P}_E . The automaton \mathcal{P}_E is deduced from the sets $First$, $Last$ and $Follow$ as follows.

Definition 3 (position automaton) *The position automaton of E , $\mathcal{P}_E = (Q, \Sigma, \delta, I, F)$, is defined by:*

- $Q = Pos(E) \cup \{0\}$, where 0 is not in $Pos(E)$,
- $I = \{0\}$,
- $F = \begin{cases} Last(E) & \text{if } Null(E) = \emptyset \\ Last(E) \cup \{0\} & \text{otherwise} \end{cases}$
- $\delta(0, a) = \{x \in First(E) \mid h(x) = a\}$, $\forall a \in \Sigma$,
- $\delta(x, a) = \{y \mid y \in Follow(E, x) \text{ and } h(y) = a\}$, $\forall x \in Pos(E)$, $\forall a \in \Sigma$.

Proposition 3 (Glushkov [4], McNaughton and Yamada [11]) *The position automaton of a regular expression E recognizes the language $L(E)$.*

Example 1 *Consider the regular expression $E = ((x^*y)^* + x(x^*y)^*y)^*$ and the linear expression $\overline{E} = ((x_1^*y_2)^* + x_3(x_4^*y_5)^*y_6)^*$. We have:*

- $Null(E) = \{\varepsilon\}$,
- $First(E) = \{x_1, y_2, x_3\}$,
- $Last(E) = \{y_2, y_6\}$,
- $Follow(E, x_1) = \{x_1, y_2\}$, $Follow(E, y_2) = \{x_1, y_2, x_3\}$,
 $Follow(E, x_3) = Follow(E, y_5) = \{x_4, y_5, y_6\}$,
 $Follow(E, x_4) = \{x_4, y_5\}$, $Follow(E, y_6) = \{x_1, y_2, x_3\}$.

The position automaton of E is shown by the Figure 1.

5.2 Homogeneity of position automata

Lemma 5 *The position automaton \mathcal{P}_E of a regular expression E is homogeneous.*

Proof. From Definition 3, we deduce that:

1. There is no transition in state 0 , since 0 is not a position of E .

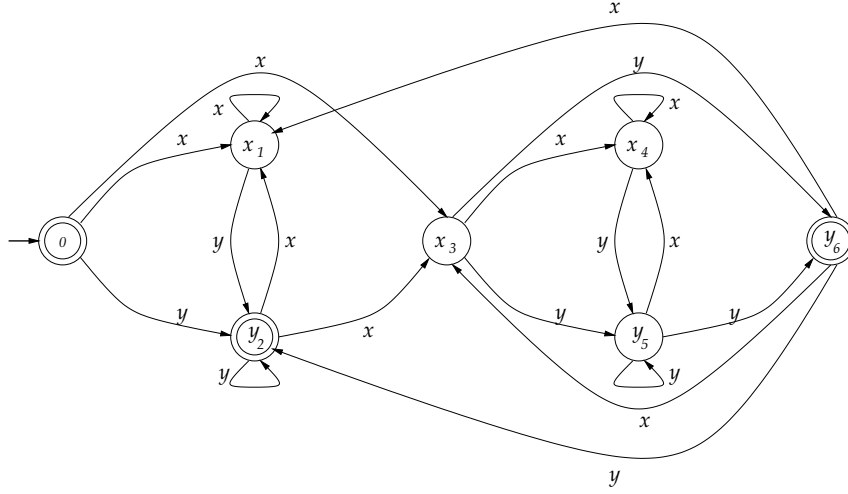


Figure 1: The position automaton for $E = ((x^*y)^* + x(x^*y)y)^*$.

2. By definition of the transition function δ , for all x in Q , for all y in $Pos(E)$, and for all a in Σ , we have: $y \in \delta(x, a) \Rightarrow h(y) = a$. Consequently, for all y in $Pos(E)$, all transitions in y have the same label $h(y)$.

■

5.3 Homogeneous bounds for position automata

By Lemma 5, complexity bounds of Theorem 2 apply to position automata. On the other hand, a position automaton is related to a regular expression, and it has a unique initial state, with no in-transition. Theorem 2 is therefore restated according to these specific features.

Theorem 3 *Let E be a regular expression over the alphabet Σ . Let n_a be the number of occurrences of symbol a in E , and $n = 1 + \sum_{a \in \Sigma} n_a$. Let \mathcal{P}_E be the position automaton of E , and \mathcal{D}_E be the subset-automaton of \mathcal{P}_E . Let n' be the number of states in \mathcal{D}_E . The automaton \mathcal{D}_E can be computed with the following complexity:*

1. Upper bound on the number of states: $n' \leq (\sum_{a \in \Sigma} 2^{n_a}) - |\Sigma| + 1$.
2. Transition computation:
 - (a) $O(n_\ell \times n_a)$ for the transition $\delta'(q', a)$ from a non-initial state q' labeled with symbol ℓ , for all a in Σ ,
 - (b) $O(n_\ell \times n)$ for the set of transitions $\delta'(q')$ from a non-initial state q' labeled with symbol ℓ ,
 - (c) $O(n \times \sum_{\ell \in \Sigma} n_\ell 2^{n_\ell})$ for the set of transitions of the subset-automaton.

3. Set equality testing:

(a) $O(\sum_{a \in \Sigma} \sqrt{n_a} 2^{2n_a})$ when using lists,

(b) $O(\sum_{a \in \Sigma} n_a^2 \log(n_a) 2^{n_a})$ when using balanced search trees.

Example 2 $E = (a_1 + (a_2 + b_3)^* a_4)(a_5 + b_6)^*$: the automata \mathcal{P}_E and \mathcal{D}_E are given by Figure 2; the standard bound is 127 and the new bound is 19.

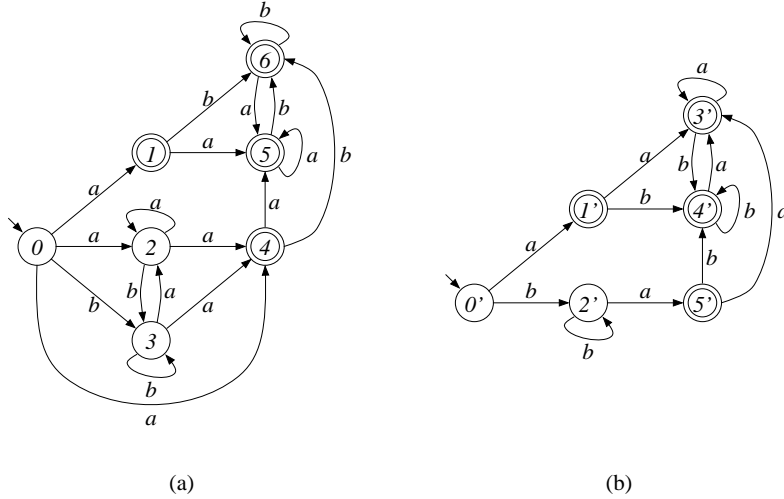


Figure 2: a. The nondeterministic automaton \mathcal{P}_E . b. The deterministic automaton \mathcal{D}_E which is the subset automaton of \mathcal{P}_E .

Example 3 $E = (a + b)^*(babab(a + b)^*bab + bba(a + b)^*bab)(a + b)^*$ [2]: the standard bound is 8.3×10^6 and the new bound is 8.7×10^3 .

6 ZPC-structures

An implicit construction of the position automaton, the so-called ZPC-structure [20, 21, 13, 14], has been developed in 1995. It can be obtained in $O(|E|)$ time, it requires $O(|E|)$ space, and it can be converted into a position automaton in $O(|E|^2)$ time. The main feature of a ZPC-structure is that, for all subset P of states, the subset $\delta(P)$, and the family of subsets $(\delta(P, a))_{a \in \Sigma}$, can be computed in $O(|E|)$ time. A consequence is that operations usually performed on a NFA, such as membership language testing or determinization, can be carried out on the implicit structure, without expanding it, and with a lower complexity.

In this section, we first recall how a regular expression can be converted into its ZPC-structure. Then we review the major properties of ZPC-structures, and deduce further improvements of subset construction.

6.1 From a regular expression to its ZPC-structure

Let us provide an unformal description of the conversion of a regular expression into its ZPC-structure. A regular expression is said *nullable* if and only if the language it denotes recognizes the empty word.

1. Start from the syntax tree $T(\overline{E})$ of \overline{E} . Process a depth-first search and link each leaf to its successor and each node to its leftmost leaf and to its rightmost leaf. Notice that these basic links give a direct access to the set of positions of a node.
2. Make two copies of $T(\overline{E})$. These copies will soon be changed into two forests respectively encoding the *Last* sets and the *First* sets of the subexpressions of E . These forests are respectively denoted by $Lasts(E)$ and $Firsts(E)$. Notice that the initial value of the basic links in $Lasts(E)$ match the case where the *Last* set of a node is the union of the *Last* sets of its children (or is the *Last* set of its child, for a unary node). A similar property holds in $Firsts(E)$.
3. The *Last* set of a concatenation node whose right child is not nullable is the *Last* set of its right child. We therefore process a depth-first search in $Lasts(E)$ and, for each concatenation node, we disable the left child connection, and update the leftmost pointer.
4. Similarly, the *First* set of a concatenation node whose left child is not nullable is the *First* set of its left child. For each concatenation node in $Firsts(E)$, we therefore disable the right child connection, and update the rightmost pointer.
5. The set of transitions produced by a concatenation operation is equal to the cartesian product of the *Last* set of its left child by the *First* set of its right child. Similarly, the set of transitions produced by a Kleene's star operation is equal to the cartesian product of the *Last* set of its child by the *First* set of its child. Such sets of transitions are called *follow sets*. For a concatenation node, we therefore connect its left child in $Lasts(E)$ to its right child in $Firsts(E)$. For a Kleene's star node, we connect its child in $Lasts(E)$ to its child in $Firsts(E)$. We call *follow link* a link representing a follow set.
6. Redundant follow links may be eliminated, so that each transition is encoded by a unique follow link.

Example 4 Let $E = a(b+a)^* + a$ and $\overline{E} = a_1(b_2+a_3)^* + a_4$. The ZPC-structure of E is shown in Figure 3.

Remark 4 It is helpful to add two special positions, $\$$ and $\#$, and to process the expression $\overline{E} = \$(a_1(b_2+a_3)^* + a_4)\#$. Position $\$$ is associated to the initial state and is involved in the follow set $\{\$\} \times First(E)$. Position $\#$ is reached from positions in $Last(E)$ by scanning the end of the input word. It only appears in the set $Last(E) \times \{\#\}$. Notice that $\$$ also appears in this set if E is nullable.

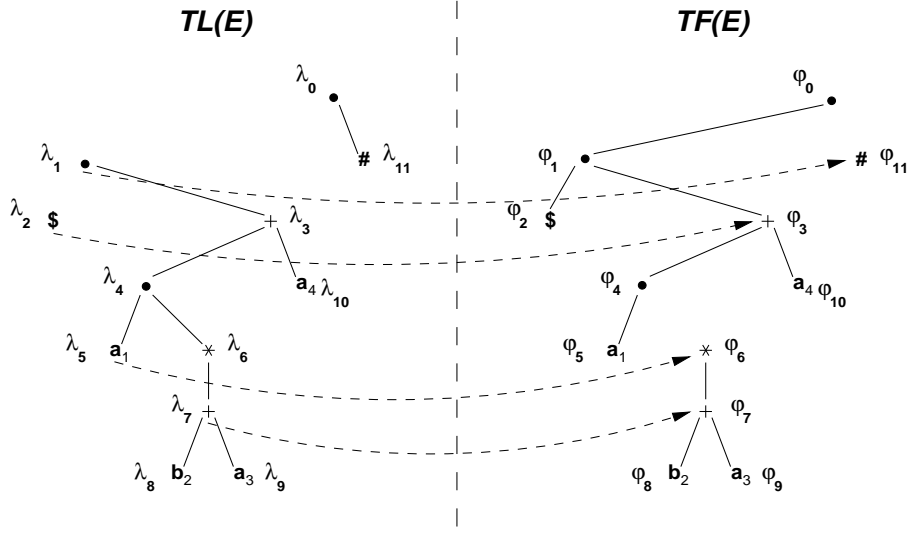


Figure 3: The ZPC-structure of $E = a(b + a)^* + a$.

Remark 5 *It is easier to describe the ZPC-structure construction using two forests. But they need not be implemented; the syntax tree, added with suitable pointers and booleans to control disabled connections, is a better representation.*

6.2 ZPC-structure properties

Formal properties of the ZPC-structure are reviewed now. Extended proofs can be found in [20, 13].

Proposition 4 ([20, 13]) *Let E be a regular expression and \mathcal{S}_E be its ZPC-structure. The following properties hold:*

1. *The structure \mathcal{S}_E can be constructed in $O(|E|)$ space and time.*
2. *The structure \mathcal{S}_E can be converted into the position automaton of E in $O(|E|^2)$ time.*
3. *The structure \mathcal{S}_E allows an $O(|E|)$ space and time computation of the following sets:*
 - (a) *the set Λ of nodes in $\text{Lasts}(E)$ whose Last sets intersect a given position set X , and which are the origin of a follow link,*
 - (b) *the set $Y = \bigcup_{\varphi \in \Phi} \text{First}(\varphi)$ of positions in the First set of the target φ of follow links belonging to a given set.*

Let X be a position subset and Y be the set of positions which are a target for a transition exiting from a position in X . We have: $Y = \delta(X) = \bigcup_{x \in X} \delta(x)$.

A fast algorithm to compute Y can be deduced from Proposition 4.3. A detailed exposition can be found in [15], where membership testing is investigated.

Proposition 5 *Given a position subset X , the set $Y = \delta(X)$ can be computed in $O(|E|)$ space and time according to the following algorithm:*

1. Compute the set Λ of nodes λ in $Lasts(E)$ such that $Last(\lambda) \cap X \neq \emptyset$ and there exists a follow link exiting from node λ .
2. Compute the set Φ of nodes φ in $Firsts(E)$ such that there exists a follow link in Λ entering in φ . The set $Y = \delta(X)$ is such that $Y = \bigcup_{\varphi \in \Phi} First(\varphi)$.
3. Deduce a set Φ' from Φ so that the set Y be computed according to the formula: $Y = \biguplus_{\varphi \in \Phi'} First(\varphi)$.

Example 5

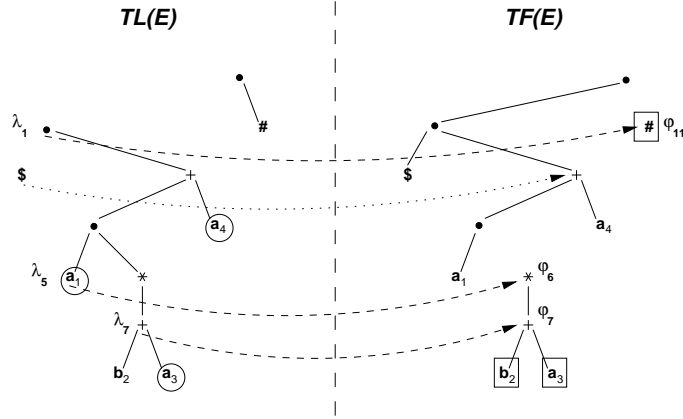


Figure 4: ZPC-structure of $\overline{E} = a_1(b_2 + a_3)^* + a_4$.

step 1	$X = \{1, 3, 4\}$	\rightarrow	$\Lambda = \{\lambda_1, \lambda_5, \lambda_7\}$
step 2	$\Lambda = \{\lambda_1, \lambda_5, \lambda_7\}$	\rightarrow	$\Phi = \{\varphi_{11}, \varphi_6, \varphi_7\}$
step 3	$\Phi = \{\varphi_{11}, \varphi_6, \varphi_7\}$	\rightarrow	$\Phi' = \{\varphi_{11}, \varphi_6, \varphi_7\} \rightarrow Y = \{2, 3, \#\}$

6.3 Transition computation

Let q' be a deterministic state. According to the algorithm of Proposition 5, we compute the sets $\Lambda(q')$ and $\Phi'(q')$, associated to the subset q' , and the set $\delta(q') = \biguplus_{\varphi \in \Phi'(q')} First(\varphi)$. The set $\delta(q')$ can be obtained in $O(|E|)$ space and time. By Lemma 4, subsets $(\delta'(q', a))_{a \in \Sigma}$ can be retrieved from $\delta(q')$ set with an overall $O(|Q|)$ time complexity, according to the formula: $\delta'(q', a) = \delta(q') \cap Q'_a$. Since the number n of states in the position automaton of E is equal to $\|E\| + 1$, and since we can assume that $O(|E|) = O(\|E\|)$, the following proposition holds:

Proposition 6 *Using the ZPC-structure of E , the time complexity of the transition computation step is the following:*

1. $O(n)$ for the set of transitions from a state q' ,
2. $O(n \sum_{\ell \in \Sigma} 2^{n_\ell})$ for the set of transitions of the subset-automaton.

6.4 Set equality testing

Our aim is to reduce the size of subsets involved in set equality testing. We show that for all q' in Q' , a set $\gamma(q')$ of nodes in $T(E)$ can be deduced from the set $\Phi'(q')$, with the following properties:

1. $\delta(q') = \delta(r') \Leftrightarrow \gamma(q') = \gamma(r')$,
2. $|\gamma(q')| \leq |\delta(q')|$,
3. The set $\gamma(q')$ is obtained in $O(|E|)$ space and time.

Let us first notice that properties (2) and (3) are satisfied by the set $\Phi'(q')$, which is such that: $\delta(q') = \bigsqcup_{\varphi \in \Phi'(q')} First(\varphi)$. The number of *First* sets which occur in the disjoint union is obviously less than or equal to the number of positions in $\delta(q')$, hence property (2) is satisfied. Property (3) comes from Proposition 4. Property (1) is not satisfied however, since we can have: $\Phi'(q') \neq \Phi'(r')$ for two distinct states q' and r' such that $\delta(q') = \delta(r')$. This situation is illustrated by the following example:

Example 6 *We consider the Example 5 (Figure 4). After determinization, we obtain the following results:*

$$\begin{array}{llll} 0' \rightarrow \{0\} & \delta(0') = \{1, 4\} & \delta(0', a) = \{1, 4\} & \delta(0', b) = \emptyset \\ 1' \rightarrow \{1, 4\} & \delta(1') = \{2, 3, \#\} & \delta(1', a) = \{3\} & \delta(1', b) = \{2\} \\ 2' \rightarrow \{3\} & \delta(2') = \{2, 3, \#\} & & \\ 3' \rightarrow \{2\} & \delta(3') = \{2, 3, \#\} & & \end{array}$$

It implies that $\delta(1') = \delta(2') = \{2, 3, \#\}$ whereas $\{\varphi_6, \varphi_{11}\} = \Phi'(1') \neq \Phi'(2') = \{\varphi_7, \varphi_{11}\}$.

Let us now consider the set $\gamma(q')$ defined as follows:

Definition 4 *Let q' be a deterministic state. The set $\gamma(q')$ is the smallest set of nodes in $T(E)$ such that: $\delta(q') = \bigcup_{\varphi \in \gamma(q')} Pos(\varphi)$.*

For all φ in $\Phi'(q')$, let $a(\varphi)$ be the farthest parent of φ in $T(E)$ such that $First(\varphi) \subseteq Pos(a(\varphi)) \subseteq \delta(q')$. Two distinct nodes φ_1 and φ_2 in $\Phi'(q')$ may be such that $a(\varphi_1) = a(\varphi_2)$. Furthermore, substituting a node φ in $\Phi'(q')$ by $a(\varphi)$ adds no new position in the set $\delta(q')$. It implies that the set $\gamma(q')$ can be deduced from $\Phi'(q')$ by substituting each node φ by the node $a(\varphi)$.

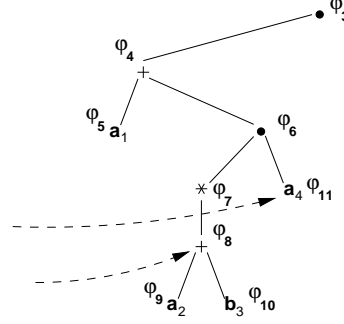


Figure 5: Computation of $\gamma(q')$.

Example 7 In this example, we assume that $\Phi'(q') = \{\varphi_8, \varphi_{11}\}$ and $\delta(q') = \{a_2, b_3, a_4\}$. We have: $Pos(\varphi_6) = First(\varphi_8) \cup First(\varphi_{11})$. Furthermore, φ_6 is a child of φ_4 which is such that: $Pos(\varphi_4) \not\subseteq \delta(q')$. Thus we get: $a(\varphi_8) = a(\varphi_{11}) = \varphi_6$ and $\gamma(q') = \{\varphi_6\}$.

Proposition 7 The following properties hold:

1. $\delta(q') = \delta(r') \Leftrightarrow \gamma(q') = \gamma(r')$,
2. $|\gamma(q')| \leq |\delta(q')|$,
3. The set $\gamma(q')$ is obtained in $O(|E|)$ space and time.

Proof. For the direct part of property (1), proof comes from unicity of the set $\gamma(q')$. The inverse part of property (1) and property (2) are obvious. As for property (3), an $O(|E|)$ space and time algorithm is provided to compute the set $\gamma(q')$ (see Figure 6). ■

Finally, for all a in Σ , the set $\gamma_a(q')$ can be deduced from the set $\gamma(q')$ in the following way: $\gamma_a(q') = \{\varphi \in \gamma(q') \mid Pos(\varphi) \cap Q_a \neq \emptyset\}$. The following proposition deduces from Proposition 7:

Proposition 8 The following properties hold:

1. $\delta(q', a) = \bigcup_{\varphi \in \gamma_a(q')} Pos(\varphi)$,
2. $\delta(q', a) = \delta(r', a) \Leftrightarrow \gamma_a(q') = \gamma_a(r')$,
3. $|\gamma_a(q')| \leq |\delta(q', a)|$,
4. The set $\gamma_a(q')$ is obtained in $O(|E|)$ space and time.

Since sets $\gamma_a(q')$ and $\delta(q', a)$ are computed with the same complexity and since $\gamma_a(q')$ is smaller than $\delta(q', a)$, using sets $\gamma_a(q')$ instead of sets $\delta(q', a)$ in the set equality testing step should speed up subset construction.

6.5 ZPC-structure complexity

Complexity results about ZPC-structures are summarized in the following theorem:

Theorem 4 *Let E be a regular expression over the alphabet Σ . Let n_a be the number of occurrences of symbol a in E , and $n = 1 + \sum_{a \in \Sigma} n_a$. Let \mathcal{P}_E be the position automaton of E , and \mathcal{D}_E be the subset-automaton of \mathcal{P}_E . Let n' be the number of states in \mathcal{D}_E . The automaton \mathcal{D}_E can be deduced from the ZPC-structure \mathcal{S}_E of E with the following complexity:*

1. *Upper bound on the number of states: $n' \leq (\sum_{a \in \Sigma} 2^{n_a}) - |\Sigma| + 1$.*
2. *Transition computation:*
 - (a) *$O(n)$ for the set of transitions from a state q' ,*
 - (b) *$O(n \times \sum_{a \in \Sigma} 2^{n_a})$ for the set of transitions of the subset-automaton.*
3. *Set equality testing: the theoretical complexity is the same as for position automata; using sets $\gamma_a(q')$ should however speed up the computation.*
 - (a) *$O(\sum_{a \in \Sigma} \sqrt{n_a} 2^{2n_a})$ when using lists,*
 - (b) *$O(\sum_{a \in \Sigma} n_a^2 \log(n_a) 2^{n_a})$ when using balanced search trees.*

7 Conclusion

The homogeneity property significantly reduces the number of deterministic states; as a consequence, the complexity of the transition computation and of the set equality testing is generally much lower for homogeneous automata than for unrestricted automata. This complexity can be lessened still further in the case of position automata as far as computation is carried out on the ZPC-structure whose main feature is to allow the computation of transition sets via disjunctive unions.

Acknowledgements

I would like to thank Derick Wood for his constructive review of a first version of this paper. I am deeply grateful to Evelyne for a valuable linguistic assistance.

References

- [1] A. V. Aho, J. E. Hopcroft, and J. D. Ullman. *The design and analysis of computer algorithms*. Addison-Wesley, Reading, MA, 1974.
- [2] V. Antimirov. Partial derivatives of regular expressions and finite automaton constructions. *Theoret. Comput. Sci.*, 155:291–319, 1996.

- [3] J.-M. Champarnaud and G. Hansel. Automate, a computing package for automata and finite semigroups. *J. Symbolic Comput.*, 12:197–220, 1991.
- [4] V. M. Glushkov. The abstract theory of automata. *Russian Mathematical Surveys*, 16:1–53, 1961.
- [5] J. E. Hopcroft and J. D. Ullman. *Introduction to Automata Theory, Languages and Computation*. Addison-Wesley, Reading, MA, 1979.
- [6] J. H. Johnson. A program for computing finite automata. Unpublished Report, University of Waterloo, Canada, 1986.
- [7] J. H. Johnson and D. Wood. Instruction computation in subset construction. In D. Raymond, D. Wood, and S. Yu, editors, *Automata Implementation : First International Workshop on Implementing Automata, WIA'96*, number 1260 in Lecture Notes in Computer Science, pages 64–71, London, Ontario, 1997. Springer, Berlin.
- [8] S. Kleene. Representation of events in nerve nets and finite automata. *Automata Studies*, Ann. Math. Studies 34:3–41, 1956. Princeton U. Press.
- [9] T. K. S. Leslie. Efficient approaches to subset construction. Technical Report CS-92-29, Department of Computer Science, University of Waterloo, Waterloo, Ontario, Canada, 1992.
- [10] T. K. S. Leslie, D. R. Raymond, and D. Wood. The expected performance of subset construction. non publié, 1996.
- [11] R. F. McNaughton and H. Yamada. Regular expressions and state graphs for automata. *IEEE Transactions on Electronic Computers*, 9:39–57, March 1960.
- [12] D. Perrin. Finite automata. In J. van Leeuwen, editor, *Handbook of Theoretical Computer Science, Formal Models and Semantics*, volume B, pages 1–57. Elsevier, Amsterdam, 1990.
- [13] J.-L. Ponty, D. Ziadi, and J.-M. Champarnaud. A new quadratic algorithm to convert a regular expression into an automaton. In D. Raymond, D. Wood, and S. Yu, editors, *Automata Implementation : First International Workshop on Implementing Automata, WIA'96*, number 1260 in Lecture Notes in Computer Science, pages 109–119, London, Ontario, 1997. Springer, Berlin.
- [14] J.-L. Ponty. Algorithmique et implémentation des automates. Thèse, Université de Rouen, France, 1997.
- [15] J.-L. Ponty. An efficient null-free procedure for deciding regular language membership. *Theoret. Comput. Sci.*, WIA'97 Special Issue, D. Wood and S. Yu, editors, to appear.

- [16] M. O. Rabin and D. Scott. Finite automata and their decision problems. *IBM J. Res.*, 3(2):115–125, 1959.
- [17] D. Raymond and D. Wood. Grail, a C++ library for automata and expressions. *J. Symbolic Comput.*, 17:341–350, 1994.
- [18] D. Wood. *Theory of Computation*. Wiley, New York, 1987.
- [19] S. Yu. Regular languages. In G. Rozenberg and A. Salomaa, editors, *Handbook of Formal Languages*, volume I, Word, Language, Grammar, pages 41–110. Springer, Berlin, 1997.
- [20] D. Ziadi, J.-L. Ponty and J.-M. Champarnaud. Passage d’une expression rationnelle à un automate fini non-déterministe, Journées Montoises (1995), *Bull. Belg. Math. Soc.*, 4:177-203, 1997.
- [21] D. Ziadi. Algorithmique parallèle et séquentielle des automates. Thèse, Université de Rouen, France, 1996.

```

procedure create- $\gamma(\Phi', Firsts(E), \gamma)$ 
  /*  $\Phi'$  and  $Firsts(E)$  are input parameters */
  /*  $\gamma$  is an output parameter */
  /*  $d$  is a local boolean array */
  function dans( $\varphi$ , in)
    /*  $\varphi$  is a node of  $T(E)$  and is used as an input parameter */
    /* in is a boolean array and is used as an input/output parameter */
    begin
      if (in[ $\varphi$ ] = false) or ( $\varphi \notin Pos(E)$ ) then
        switch  $\varphi$ 
          case '.,'+ : in[ $\varphi$ ]  $\leftarrow$  (dans(left( $\varphi$ ), in)  $\wedge$  dans(right( $\varphi$ ), in))
          case '*': in[ $\varphi$ ]  $\leftarrow$  dans(child( $\varphi$ ), in)
        end switch
      return in[ $\varphi$ ]
    end
  procedure traversal( $\varphi$ ,  $d$ ,  $\gamma$ )
    /*  $\varphi$  is a node of  $T(E)$  and is used as an input parameter */
    /*  $\gamma$  is an output parameter */
    begin
      if  $d[\varphi]$  then  $\gamma \leftarrow \gamma \cup \{\varphi\}$ 
      else if  $\varphi \notin Pos(E)$ 
        then
          begin
            traversal(left( $\varphi$ ),  $\gamma$ )
            traversal(right( $\varphi$ ),  $\gamma$ )
          end
        end
    end
  begin
    foreach  $\varphi \in Firsts(E)$  do  $d[\varphi] \leftarrow$  false
    foreach  $\varphi \in \Phi'$  do if parent( $\varphi$ ) = '*' then  $d[\varphi] \leftarrow$  true
    dans(rac( $T(E)$ ),  $d$ ); traversal(rac( $T(E)$ ),  $d$ ,  $\gamma$ )
  end

```

Figure 6: Algorithm to compute the set $\gamma(q')$.