Foundations of Software Technology and Theoretical Computer Science (Bangalore) 2008.
Editors: R. Hariharan, M. Mukund, V. Vinay; pp -

An Optimal Construction of Finite Automata from Regular Expressions

Stefan Gulan, Henning Fernau
Universität Trier
{gulan,fernau}@uni-trier.de

ABSTRACT. We consider the construction of finite automata from their corresponding regular expressions by a series of digraph-transformations along the expression's structure. Each intermediate graph represents an extended finite automaton accepting the same language. The character of our construction allows a fine-grained analysis of the emerging automaton's size, eventually leading to an optimality result.

1 Introduction

Regular expressions provide a description of regular languages in a manner convenient for the human reader. On the machine level, however, the most appropriate representation is arguably that of finite automata. Thus, considerable effort has been put into ways of constructing automata describing the same language as a given expression. All algorithms known to the authors work by either incorporating the expression's syntactic structure into the state graph of the emerging automaton [OF61, Kle65, Tho68, SSS88, IY03] or by looking for first-time occurrences of symbols in subexpressions [Glu61, MY60, BS86]. The first kind of construction generally results in an NFA with ϵ-transitions (ϵNFA, for short), the latter produces no such transitions and may even provide a DFA. An exhaustive overview is given in [Wat94].

Our construction yields an ϵNFA. No tight bound for the size of such an automaton representing a given expression has been published yet. Ilie & Yu [IY03] came pretty close, proving a lower bound of $\frac{4}{3}$ times the size of a given expression while constructing an ϵNFA smaller than $\frac{3}{2}$ times the expression length. We close this gap by raising the lower bound and giving a construction reaching that bound in the worst case. Note, however, that plenty of definitions of the sizes of automata and regular expressions are afloat, some of which are compared in [EKSW05]. For comparability, we stick by the definition given in [IY03].

The algorithm presented in this paper is basically an extension to the one given in [OF61], which is, together with a variation of Thompson's algorithm in [Wat94], the only top-down algorithm among a variety of bottom-up procedures. It turns out that the top-down character is very helpful in the analysis, since it allows systematic construction of an expression yielding the worst ratio of automaton-to-expression sizes. This construction relies on extremal combinatorial arguments for inferring structural properties of a worst-case input. To our knowledge this is a novel approach to this kind of problem.

2 Preliminaries

Enclosing braces for singleton sets will be omitted. Let $\mathcal{A}$ be a finite set of symbols, called *alphabet*, the elements of $\mathcal{A} \cup \epsilon$ will be called *literals*. The set of regular expressions over $\mathcal{A}$, denoted $Reg(\mathcal{A})$, is the closure of $\mathcal{A} \cup \epsilon$ under product $\bullet$, sum $+$ and Kleene-star *. Operator precedence is $*, \bullet, +$. We will casually speak of *expressions* only. In the following, α and β will always be expressions. The regular language expressed by α is denoted $L(\alpha)$. We will call α and β *equivalent*, denoted $\alpha \equiv \beta$, if $L(\alpha)=L(\beta)$. The number of products (sums, stars) in α will be denoted $|\alpha|_\bullet$ ($|\alpha|_+$, $|\alpha|_*$). Likewise, the number of literals in α, counted with multiplicity, will be denoted $|\alpha|_{\mathcal{A}}$. The *size* of an expression is defined as $|\alpha|:=|\alpha|_\bullet+|\alpha|_++|\alpha|_*+|\alpha|_{\mathcal{A}}$. We call α *complex*, if $|\alpha| \geq 2$. The set of subexpressions of α will be denoted $sub(\alpha)$.

Both iterated products and sums will be denoted as is common in arithmetic, defining

$$\prod_{i=1}^{n} \alpha_i := \alpha_1 \bullet \alpha_2 \bullet \ldots \bullet \alpha_n \quad \text{and} \quad \sum_{i=1}^{n} \alpha_i := \alpha_1 + \alpha_2 + \ldots + \alpha_n$$

Each α_i as above will be called an *operand* to the product or sum. An iterated product (sum) which is not operand to a product (sum) itself, will be called *maximal*. If all operands in a maximal product (sum) are starred, it will be called *star-maximal*.

An *extended finite automaton*, short *EFA*, is a 5-tuple $E=(Q,\mathcal{A},\delta,q_0,F)$, where $q_0 \in Q$, $F \subseteq Q$, and $\delta \subset Q \times Reg(\mathcal{A}) \times Q$. This renders conventional FAs a special case of EFAs. An EFA is called *normalized*, if $|F|=1$. A pair $(q,w) \in Q \times \mathcal{A}^*$ is called *configuration* of E, valid changes in E's configuration are denoted by $\vdash$, writing $(q,vw) \vdash (q',w)$ if $(q,\alpha,q') \in \delta$ and $v \in L(\alpha)$. The language accepted by E is $L(E)=\{w | (q_0,w) \vdash^* (q_f,\epsilon), q_f \in F\}$, where $\vdash^*$ is the reflexive-transitive closure of $\vdash$.

The class of regular languages is not extended by allowing regular expressions as labels in automata, see [Woo87] for a proper introduction. The size of an EFA E is $|E|:=|Q|+|\delta|$. The sets of transitions leaving and reaching some $q \in Q$ are given by $q^+:=\delta \cap (q \times Reg(\mathcal{A}) \times Q)$ and $q^-:=\delta \cap (Q \times Reg(\mathcal{A}) \times q)$, respectively. A set of transitions $\gamma = \{(q_i,\alpha_i,q_{i+1}) | 1 \leq i \leq n-1\} \cup (q_n,\alpha_n,q_1)$ is called *cycle*.

Let A be a FA generated from α by some algorithm $\mathcal{C}$. We call $\frac{|A|}{|\alpha|}$ the *conversion-ratio* of $\mathcal{C}$ with respect to α. The maximal conversion-ratio of $\mathcal{C}$ with respect to any expression, will simply be called conversion-ratio of $\mathcal{C}$. An expression reaching this bound is said to be *worst-case*.

3 A Lower Bound

First we improve on a lower bound for *any* construction of FAs from expressions, given by Ilie & Yu in [IY03], by a slight variation of their argument. To this end, a property of digraphs is shown, in which we refer to both vertices and arcs as *elements*.

PROPOSITION 1. *Consider a digraph (V,A). Let L,R be nonempty, disjoint subsets of V such that*

1. *there is a path from each $l \in L$ to each $r \in R$,*
2. *there is no path connecting any two vertices $l,l' \in L$ or any $r,r' \in R$.*

Then at least $\min\{|L||R|,|L|+|R|+1\}$ *elements are necessary to realize these paths.*

PROOF. Two cases need to be considered:

1. There is no vertex on any path connecting l with r. This can only be realized with $|L||R|$ arcs, by pairwise connections.
2. There is at least one vertex b on a path connecting $l_b \in L$ with $r_b \in R$, this path contains at least 3 elements. To connect l_b with the vertices of $R \backslash r_b$ at least $|R|-1$ further arcs are necessary. An additional $|L|-1$ arcs are leaving the vertices of $L \backslash l_b$. These numbers total to $|L|+|R|+1$.

Next we show the actual lower bound. Both states and transitions of an FA A will be called elements, the number of elements is simply $|A|$.

THEOREM 2. *Let $x_{i,j}$ be distinct literals, consider the expression*

$$\begin{aligned}\alpha &= \prod_{i=1}^{n}(x^*_{2i-1,1}+x^*_{2i-1,2})(x^*_{2i,1}+x^*_{2i,2}+x^*_{2i,3})\\ &= (x^*_{1,1}+x^*_{1,2})(x^*_{2,1}+x^*_{2,2}+x^*_{2,3})\ldots(x^*_{2n-1,1}+x^*_{2n-1,2})(x^*_{2n,1}+x^*_{2n,2}+x^*_{2n,3})\end{aligned}$$

Any normalized automaton A satisfying $L(A) = L(\alpha)$ has at least size $22n+1$.

PROOF. In A, each $x_{i,j}$ is read on some cycle $\gamma_{i,j}$ comprising at least one transition incident to a state $q_{i,j}$, i.e., 2 elements. The $\gamma_{i,j}$ are disjoint, since literals of the same factor occur mutually exclusive and literals of different factors are ordered by α. Thus $5n$ cycles, accounting for at least $10n$ elements, are required. As for the connectivity of cycles, no path may lead from $\gamma_{i,j}$ to $\gamma_{i,k}$, if $j \neq k$, however, there need to be paths from $\gamma_{i,j}$ to $\gamma_{i+1,k}$. This carries over to the connectivity of the $q_{i,j}$, thus each two sets of states $q_{i,j}$ and $q_{i+1,j'}$ satisfy the conditions given in Prop. 1. Since one of the sets contains 2, the other one 3 states, by Prop. 1 at least 6 Elements are needed to ensure connectivity. As there are $2n-1$ such pairs, $12n-6$ elements are needed to connect them. This totals to $22n-6$ elements, additionally, 2 states and 5 transitions are necessary to ensure a normalized FA.

For the following, note that α from Thm. 2 has size $15n-1$.

COROLLARY 3. *The conversion-ratio of any algorithm converting expressions to normalized FAs is bounded from below by*

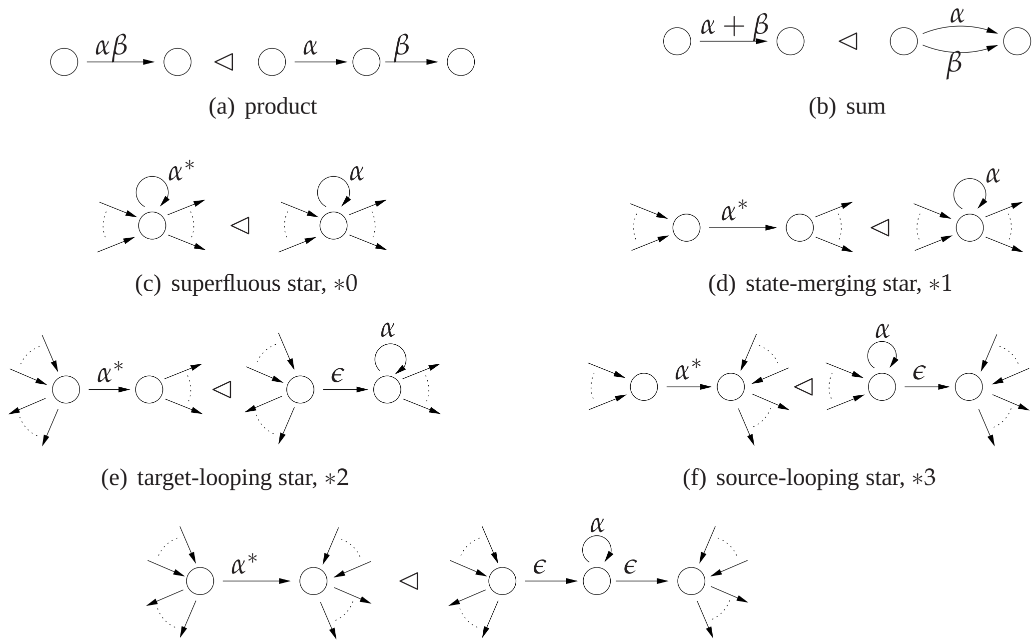
$$\frac{|A|}{|\alpha|} \geq \frac{22n+1}{15n-1} > \frac{22}{15} + \frac{1}{|\alpha|} = 1.4\bar{6} + \frac{1}{|\alpha|}$$

4 Construction

The idea is to expand an initial EFA according to the structure of the expression, by introducing as few states and transitions as possible, while decomposing transition labels. Certain substructures in the expanded automata will be replaced by smaller equivalents. This is done until an ϵNFA emerges, i.e., there are no more complex labels.

DEFINITION 4._[Expansion] Let $E = (Q, \mathcal{A}, \delta, q_0, F)$ be an EFA with a complex labeled transition t. We call an EFA $E' = (Q', \mathcal{A}, \delta', q_0, F)$ the_ expansion _of E, if it is derived from E according to the label of t as follows:_

- *if* $t = (p, \alpha\beta, q)$ *then* $Q' = Q\dot{\cup}p'$, $\delta' = \delta \setminus t \cup \{(p, \alpha, p'), (p', \beta, q)\}$
- *if* $t = (p, \alpha+\beta, q)$ *then* $Q' = Q$, $\delta' = \delta \setminus t \cup \{(p, \alpha, q), (p, \beta, q)\}$
- *if* $t = (p, \alpha^*, q)$, *we distinguish several cases*
  - $*0$*: if* $p = q$, *replace* α^* *with* α,
    let $Q' = Q$, $\delta' = \delta \setminus t \cup (q, \alpha, q)$
  - $*1$*: if* $|p^+| = |q^-| = 1$, *merge* q *into* p*:*
    let $Q' = Q \setminus q$, $\delta' = \delta \setminus (q^+ \cup q^-) \cup \{(p, \gamma, r) | (q, \gamma, r) \in \delta\} \cup (p, \alpha, p)$
  - $*2$*: if* $|p^+| > 1$, $|q^-| = 1$, *introduce a loop in* q*:*
    let $Q' = Q$, $\delta' = \delta \setminus t \cup \{(p, \epsilon, q), (q, \alpha, q)\}$



(a) product (b) sum

(c) superfluous star, $*0$ (d) state-merging star, $*1$

(e) target-looping star, $*2$ (f) source-looping star, $*3$

(g) state-introducing star, $*4$

Figure 1: Expansions of complex labeled transitions.

$*3$: *if* $|p^+| = 1$, $|q^-| > 1$, *introduce a loop in* p:
let $Q' = Q$, $\delta' = \delta \setminus t \cup \{(p, \alpha, p), (p, \epsilon, q)\}$

$*4$: *if* $|p^+| > 1$, $|q^-| > 1$, *introduce a new state* p':
let $Q' = Q \dot{\cup} p'$, $\delta' = \delta \setminus t \cup \{(p, \epsilon, p'), (p', \alpha, p'), (p', \epsilon, q)\}$

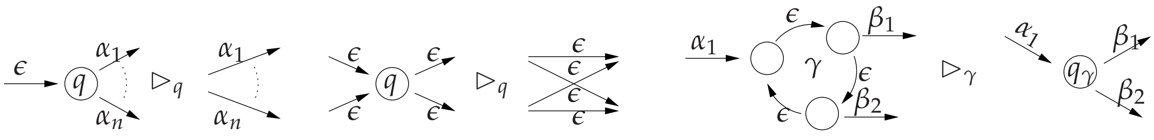
Cases are sketched in Fig. 1. Expansions will be denoted relational, writing $E \lhd_t E'$ if E' results from expansion of t in E. Occasionally we write $\lhd_\bullet, \lhd_+, \lhd_{*i}$ to indicate which case of Def. 4 is applied, or simply $E \lhd E'$, if both t and the case are irrelevant. The latter might be formalized as $\lhd = \lhd_\bullet \cup \lhd_+ \cup \bigcup_{0 \leq i \leq 4} \lhd_{*i}$. The n-fold iteration of $\lhd$ will be denoted $\lhd^n$, thus if $E \lhd^n E'$ there is a series of EFAs E_i, $0 \leq i \leq n$, such that $E = E_0$, $E_i \lhd E_{i+1}$, $E_n = E'$. Usually we refer to $\lhd_{(q,\alpha,q')}$ by mentioning α's operator, e.g, '$\bullet$-expansion'. Distinct $*$-expansions will be referred to as '$*0$-expansion' to '$*4$-expansion' according to Def. 4.

DEFINITION 5.*[Primal EFA] Let* $\mathcal{A}$ *be the least alphabet satisfying* $\alpha \in Reg(\mathcal{A})$. *The EFA* $A^0_\alpha = (\{q_0, q_f\}, \mathcal{A}, (q_0, \alpha, q_f), q_0, q_f\})$ *is called the* primal EFA *representing* α. *We denote by* A^i_α *any automaton satisfying* $A^0_\alpha \lhd^i A^i_\alpha$.

Thus, A^i_α denotes any EFA derived from the primal automaton representing α in a series of i expansions. Note that generally, A^i_α is not unique. However, a most useful property of $\lhd$ is that the order of expansion is irrelevant, or formally:

LEMMA 6. $\lhd$ *is locally confluent, i.e., if* $A \lhd A'$ *and* $A \lhd A''$, *then* $\exists A''' : A' \lhd A'''$ *and* $A'' \lhd A'''$.

PROOF. Given in the appendix.



(a) Y-Type, schematic (b) X-Type, schematic (c) Cycle-elimination, exemplary

Figure 2: State-eliminations (a,b) and cycle-elimination (c)

	$\lhd_\bullet$	$\lhd_+$	$\lhd_{*0}$	$\lhd_{*1}$	$\lhd_{*2}$, $\lhd_{*3}$	$\lhd_{*4}$	$\rhd_\gamma$	$\rhd_q$
$\Delta(\lvert Q\rvert)$	1	0	0	-1	0	1	$-(\lvert\gamma\rvert-1)$	-1
$\Delta(\lvert\delta\rvert)$	1	1	0	0	1	2	$-\lvert\gamma\rvert$	-1 or 0

Table 1: Number of elements introduced (i.e., removed, if negative) upon expansion and elimination, broken down to states and transitions.

COROLLARY 7. $\lhd$ *is confluent.*

PROOF. Since $\lhd$ is terminating, the claim follows from Lem. 6. Detailed proof of this argument can be found, e.g., in [Hue80].

We introduce two further conversions of different nature, altering EFAs with respect to ϵ-labeled substructures.

DEFINITION 8.*[State-Elimination] Let* $E=(Q,\mathcal{A},\delta,q_0,F)$ *be an EFA,* $q\in Q\backslash F$*. We consider two types of state-elimination, based on* q^+ *and* q^-*:*

- *Y-Type :* $q^-=(p,\epsilon,q)$, $q^+=\{(q,\alpha_1,r_1),\ldots,(q,\alpha_n,r_n)\}$.
  Then, let $\delta'=\delta\setminus(q^+\cup q^-)\cup\{(p,\alpha_1,r_1),\ldots,(p,\alpha_n,r_n)\}$
- *X-Type :* $q^-=\{(p_1,\epsilon,q),(p_2,\epsilon,q)\}$, $q^+=\{(q,\epsilon,r_1),(q,\epsilon,r_2)\}$.
  Then, let $\delta'=\delta\setminus(q^+\cup q^-)\cup\{(p_1,\epsilon,r_1),(p_1,\epsilon,r_2),(p_2,\epsilon,r_1),(p_2,\epsilon,r_2)\}$.

The q-reduct *of* E *is defined as* $E'=(Q\backslash q,\mathcal{A},\delta',q_0,F)$ *and we write* $E\rhd_q E'$.

By reverting the transitions for Y-Type elimination, a further rule—though not structurally different from the given Y-Type—is obtained.

DEFINITION 9.*[ϵCycle-Elimination] Let* $\gamma=\{(q_i,\epsilon,q_i')|1\le i\le n\}$ *be a cycle of* $E=(Q,\mathcal{A},\delta,q_0,F)$*. Let* $Q'=Q\backslash\{q_1,\ldots,q_n\}\cup q_\gamma$ *and* $\delta'=\delta\backslash\gamma\cup\{(p,\alpha,q_\gamma)|(p,\alpha,q_i)\in\delta\}\cup\{(q_\gamma,\beta,r)|(q_i,\beta,r)\in\delta\}$*. The* γ-reduct *of* E *is defined as* $E=(Q',\mathcal{A},\delta',q_0,F)$.

Note that both state- and cycle-eliminations strictly reduce the size of an EFA without re-introducing complex labels. Eliminations are illustrated in Fig.2.

Exhaustive application of expansions and eliminations to A^0_α (or any EFA, for that matter) yields an ϵNFA. A primitive algorithm is given below.

5 Analysis

Let A_α denote an ϵNFA constructed by our algorithm from A^0_α. We start by bounding $|A_\alpha|$ from above. To this end, we refine the definition of $|\alpha|_*$. Let $|\alpha|_{*i}$ denote the number of stars in α, that will be $*i$-expanded. Clearly, $|\alpha|_*=\sum_{0\le i\le 4}|\alpha|_{*i}$.

Algorithm 1 RegEx $\rightarrow \epsilon$NFA

```
A ← A_α^0
while A is not an NFA do
    choose a complex-labeled transition t in A
    let A ◁_t A'
    if ◁_t introduced some e = (q, ϵ, q') then
        if q can be eliminated then
            let A' ▷_q A''
            A' ← A''
        if q' can be eliminated then
            let A' ▷_{q'} A''
            A' ← A''
        if e is part of some ϵ-cycle γ then
            let A' ▷_γ A''
            A' ← A''
    A ← A'
end while
```

THEOREM 10. *The size of an automaton built from α by our algorithm is bounded by*

$$|A_\alpha| \leq |\alpha| + 2|\alpha|_{*4} - |\alpha|_+ + 2$$

If this bound is tight then neither state-elimination nor $*0, *1$-expansion is applied.

PROOF. A^0_α is of size 3. The number of elements introduced upon expansion is determined by $|\alpha|_\bullet, |\alpha|_+, \ldots$, weighted by the entries in Tab. 1. Using $|\alpha|_{\mathcal{A}} = |\alpha|_\bullet + |\alpha|_+ + 1$ and $|\alpha| = |\alpha|_\bullet + |\alpha|_+ + |\alpha|_{*0} + \ldots + |\alpha|_{*4} + |\alpha|_{\mathcal{A}}$, this yields:

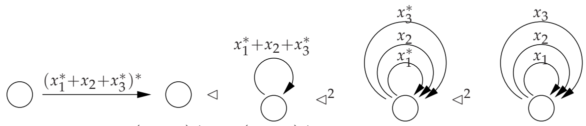
$$\begin{aligned}
|A_\alpha| &\leq 2|\alpha|_\bullet + |\alpha|_+ - |\alpha|_{*1} + |\alpha|_{*2,3} + 3|\alpha|_{*4} + 3 \\
&= |\alpha| + |\alpha|_\bullet - |\alpha|_{*0} - 2|\alpha|_{*1} + 2|\alpha|_{*4} - |\alpha|_{\mathcal{A}} + 3 \\
&\leq |\alpha| + |\alpha|_\bullet + 2|\alpha|_{*4} - |\alpha|_{\mathcal{A}} + 3 \\
&= |\alpha| + 2|\alpha|_{*4} - |\alpha|_+ + 2
\end{aligned}$$

The first inequality results from state- and ϵ-cycle eliminations, the second from $*0$- and $*1$-expansions, thus equality holds in absence of these transformations.

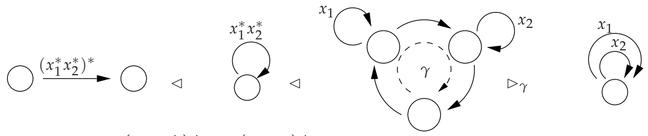
The conversion ratio of a worst-case expression can be read immediately from this term; since we will refer to this quotient rather often, we restate it explicitly in

COROLLARY 11. *Let α be worst-case, then*

$$\frac{|A_\alpha|}{|\alpha|} = 1 + \frac{2|\alpha|_{*4} - |\alpha|_+ + 2}{|\alpha|}$$



(a) $(\sum \alpha_i)^* \equiv (\sum \overline{\alpha_i})^*$ by $*0$-expansions.



(b) $(\prod \alpha_i^*)^* \equiv (\sum \alpha_i)^*$ by elimination of ϵ-cycles.

Figure 3: Transformations respect the equivalences given in Prop. 12 (ϵ-labels are omitted).

PROPOSITION 12. *Both sides in each of the following equivalences will be expanded to the same (sub)automaton:*

$$(\alpha^*)^* \equiv \alpha^* \quad \text{and} \quad (\sum \alpha_i)^* \equiv (\sum \overline{\alpha_i})^* \quad \text{and} \quad (\prod \alpha_i^*)^* \equiv (\sum \alpha_i)^*$$

where $\overline{\alpha_i} = \beta_i$, *if* $\alpha_i = \beta_i^*$ *and* α_i *otherwise.*

PROOF. The first two equivalences are realized by $*0$-expansion, the third by ϵ-cycle-elimination. Examples are given in Fig. 3.

COROLLARY 13. *Let* α *be worst-case, then* $|\alpha|_{*0} = |\alpha|_{*1} = 0$, *further both a sum with starred operands and a maximally starred product are not starred themselves.*

PROOF. By Prop. 12 we know that such sums and products would lead to $*0$-/$*1$-expansions and eliminations. Since for worst-case expressions equality in Thm. 10 holds and thus said conversions do not occur, the claim follows.

We proceed with a series of results, each putting additional constraints to the structure of a worst-case expression. Almost all proofs work by a line of argumentation that is common in extremal combinatorics: assume α is worst-case, i.e., extremal with respect to conversion-ratio, then infer some further property by contradicting extremality of α.

PROPOSITION 14. *A worst-case expression contains stars.*

PROOF. Let α be worst-case with $|\alpha|_* = 0$. Cor. 11 implies $\frac{|A_\alpha|}{|\alpha|} \leq 1 + \frac{2}{|\alpha|}$, the right-hand side of which drops below 1.4, if $|\alpha| \geq 5$. Since by Cor. 3, the conversion-ratio is bounded from below by $1.4\bar{6}$, the assumption $|\alpha|_* = 0$ is wrong, if α is worst-case.

LEMMA 15. *Let* γ^* *be a proper subexpression of* α. *Then* γ^* *will be* $*4$*-expanded iff*

- *it is operand to a sum which is not starred, or*
- *without loss of generality it occurs rightmost in a star-maximal product.*

PROOF. The first case is clear by looking at the expansion of some $\gamma^* + \beta$: If a transition labeled like this is a loop, γ^* will be $*0$-expanded, otherwise it will definitely be $*4$-expanded.

The second case is more involved: If γ^* is an infix, say, $\alpha_1 \gamma^* \alpha_2$, we distinguish 3 cases: If both α_i are non-starred, γ^* will be $*1$-expanded. If only one of the α_i is non-starred, then γ^* can be $*2$- or $*3$-expanded by introducing a loop at the state incident to the transition labeled with the non-starred α_i.

Finally, if both α_i are starred, we can by confluence assume that expansions will be applied from left to right. Then, every starred factor will be $*2$-expanded until the final one necessitates $*4$-expansion. This embraces all possible cases, giving both directions of the statement.

LEMMA 16. *Let α be worst-case, assume $\gamma^*\in sub(\alpha)$ is $*4$-expanded. Then γ^* is operand to a sum.*

PROOF. By Lem. 15, γ^* is either operand to a sum or rightmost in a star-maximal product. Assume the latter, thus $\pi = \pi_1^* \bullet \ldots \bullet \pi_{n-1}^* \bullet \gamma^*$. Construct α' from α by replacing π with $\sigma = \pi_1^* + \ldots + \pi_{n-1}^* + \gamma^*$. Then $|\alpha|=|\alpha'|$, however $2|\alpha'|_{*4} - |\alpha'|_+ = 2|\alpha|_{*4} - |\alpha|_+ + n - 1$. Since by Prop. 12 π is not starred in α, the stars in σ will not accidentally become $*0$. By Cor. 11, $\frac{|A_{\alpha'}|}{|\alpha'|} > \frac{|A_\alpha|}{|\alpha|}$, thus α is not worst-case. Therefore γ^* is necessarily operand to a sum.
The interrelation between sums and stars in a worst-case expression is further tightened in the following

LEMMA 17. *Let α be worst-case. Then*
1. *every starred subexpression in α is operand to a sum and*
2. *all operands in a maximal sum are starred.*

PROOF.
1. Assume $\gamma^*\in sub(\alpha)$ will not be $*4$-expanded. Construct α' from α by replacing γ^* with γ. Since $|\alpha'|=|\alpha|-1$, yet $|\alpha'|_{*4}=|\alpha|_{*4}$, Cor. 11 again yields $\frac{|A_{\alpha'}|}{|\alpha'|} > \frac{|A_\alpha|}{|\alpha|}$, thus α is not worst-case. Therefore each star in a worst-case expression is subject to $*4$-expansion, thus by Lem. 16 operand to a sum.
2. Let $\sum \sigma_i$ be maximal with some σ_j unstarred, i.e., a product. Construct α' from α by replacing σ_j with σ_j^*. This newly starred expressions will be $*4$-expanded (Lem. 15). Then $|\alpha'| = |\alpha|+1$, $|\alpha'|_{*4} = |\alpha|_{*4}+1$ and by Cor. 11, $|A_{\alpha'}| = |A_\alpha|+2$. Now
$$\frac{|A_{\alpha'}|}{|\alpha'|} = \frac{|A_\alpha|+2}{|\alpha|+1} > \frac{|A_\alpha|}{|\alpha|} \quad \text{iff} \quad |A_\alpha| < 2|\alpha|$$
We proceed similar to the proof of Thm. 10, additionally using that the previous item implies $|\alpha|_{*4} \le 2|\alpha|_+$:
$$\begin{aligned} |A_\alpha| &\le 2|\alpha|_\bullet + |\alpha|_+ - |\alpha|_{*1} + |\alpha|_{*2,3} + 3|\alpha|_{*4} + 3 \\ &= 2|\alpha| - |\alpha|_+ - 3|\alpha|_{*1} - |\alpha|_{*2,3} + |\alpha|_{*4} + 3 - 2|\alpha|_{\mathcal{A}} \\ &= 2|\alpha| - 2|\alpha|_+ - |\alpha|_\bullet - 3|\alpha|_{*1} - |\alpha|_{*2,3} + |\alpha|_{*4} + 2 - |\alpha|_{\mathcal{A}} \\ &\le 2|\alpha| - |\alpha|_+ - 2|\alpha|_\bullet + 1 \end{aligned}$$
By assumption, $|\alpha|_+ \ge 1$, any further binary operator pushes the right-hand side strictly below $2|\alpha'|$. Indeed, the only expression containing only one $+$ as binary operator, that reaches a conversion-ratio of 2, is $x_1^* + x_2^*$, which is of claimed structure.

LEMMA 18. *A worst-case expression α has no subexpression of the form*
$$\phi = (\prod_i \sum_j \sigma_{ij}^*)^*$$

PROOF. If $\phi \in sub(\alpha)$, ϵ-cycle elimination would occur upon expansion. By Cor. 11 then α would not be worst-case.
This allows us to provide a pretty detailed template of a worst-case expression:

LEMMA 19. *Let α be worst-case. Then the structure of α is*

$$\alpha = \prod_{i=1}^{n} \sum_{j=1}^{k_i} \sigma_{ij}^{*} \quad \textit{where} \quad \sigma_{i,j} \in \mathcal{A}$$

PROOF. By Prop. 14, a worst-case expression contains starred subexpressions, so fix some σ_{ij}^{*} which is by Lem. 17 operand to a sum. A maximal sum with stars is a factor, since it may not be starred itself and is already maximal. Further, σ_{ij} is necessarily a maximal product. If its operands were maximally starred sums, this would contradict Lem. 18, thus σ_{ij} is a product of literals. Then, σ_{ij} influences the conversion-ratio as given in Cor. 11 only by its length, which has to be minimized in order to maximize the ratio. Thus σ_{ij} is a symbol from the alphabet. From Lem. 18 it also follows that α itself may not be starred.

It remains to analyze the influence of the number of summands (the k_i in Lem. 19) on conversion-ratio. This is done in the proof of our main

THEOREM 20. *An expression α is worst-case, if its structure is*

$$\alpha = \prod_{i=1}^{n} \sum_{j=1}^{2+(i \bmod 2)} x_{ij}^{*} \quad \textit{where} \quad x_{ij} \in \mathcal{A}$$

PROOF. Let α be of the general structure given in Lem. 19, the FA produced by a series of expansions from A_{α}^{0} is sketched in Fig. 4. The sizes of these objects are

$$\begin{aligned} |\alpha| &= (n-1) + \sum_{i=1}^{n}(3k_i - 1) = 3\sum_{i=1}^{n} k_i - 1 \\ |A_{\alpha}| &= \sum_{i=1}^{n} 4k_i + n - 1 = 4\sum_{i=1}^{n} k_i + n - 1 \end{aligned}$$

thus the ratio is

$$\frac{|A_{\alpha}|}{|\alpha|} = \frac{4\sum k_i + n - 1}{3\sum k_i - 1} = 1 + \frac{\sum k_i + n}{3\sum k_i - 1}$$

The fraction on the right-hand side is maximized, if n is maximal with respect to $\sum k_i$, or equivalently, if $\sum k_i$ is minimal. Two restrictions result from prohibiting state-elimination, namely that $\forall i : k_i \geq 2$ and if $k_i{=}2$ then $k_{i-1}{>}2$ and $k_{i+1}{>}2$ (if they exist). Thus $\sum k_i$ is minimal, if k_i alternates between 2 and 3, i.e., $k_i = 2 + (i \bmod 2)$.

COROLLARY 21. *The size of an automaton produced by our construction is bounded by $\frac{22}{15}|\alpha| + 1$. The construction is optimal.*

PROOF. The value is reached by the expression given in Thm. 20, which was proven to give the maximal ratio of sizes. Since by Cor. 3 $\frac{22}{15}|\alpha| + 1$ is also a lower bound, the bound is tight, hence the construction is optimal.

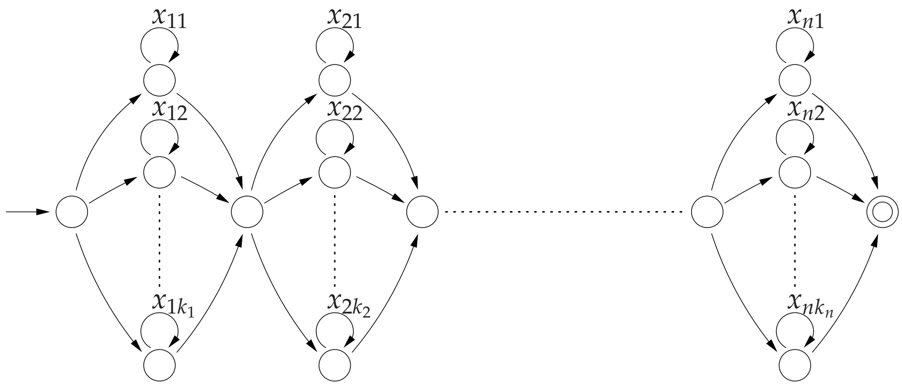


Figure 4: Automaton constructed from an expression as given in Lem. 19 (ϵ-labels are omitted).

6 Conclusions & Remarks

We have given a construction for converting regular expressions into equivalent ϵNFAs. To our knowledge it is the only provably optimal construction so far. It should be mentioned that the generated automata differ from these constructed in [IY03] only by the effects of state-elimination. This element is crucial however, both for raising the lower bound as well as for upper bound analysis as we did. On a practical detail, preprocessing the input to *reduced* expressions (as done in [IY03]) is in part realized upon execution of our algorithm.

Treatment of $\emptyset$ in expressions can easily be added to our algorithm by considering it a literal throughout the expansion/reduction-sequence and adding a final step: removing $\emptyset$-labeled transitions followed by running some reachability algorithm. The final step will reduce the size of the automaton, thus the bound is maintained even if $\emptyset$ does not count into the expressions' size. Since we consider $\emptyset$ as being of no practical relevance, it was omitted from formal treatment.
Maybe more interesting, Kleene-$+$ can be implemented by reformulating $*$-expansions, where additional ϵ-transitions need to be introduced. This yields smaller FAs than by applying the equivalence $\alpha^+ \equiv \alpha\alpha^*$ (which would double the number of elements introduced by α), yet it is not feasible with the given bound.

Finally note that the construction not unique in the general case, since state-eliminations is not confluent. This can be remedied by adding rules that take the in- and out-degrees of the states adjacent to the eliminated one into consideration, however this is not at the attention of this paper. A closer analysis will be available in a future article.

References

[BS86] Gerard Berry and Ravi Sethi. From regular expressions to deterministic automata. *Theoretical Computer Science*, 48:117–126, 1986.

[EKSW05] Keith Ellul, Bryan Krawetz, Jeffrey Shallit, and Ming-Wei Wang. Regular expressions: new results and open problems. *Journal of Automata, Languages and Combinatorics*, 10(4):407–437, 2005.

[Glu61] Victor Michailowitsch Glushkov. The abstract theory of automata. *Russian Mathematical Surveys*, 16:1–53, 1961.

[Hue80] Gérard Huet. Confluent reductions: Abstract properties and applications to term rewriting systems. *Journal of the ACM*, 27(4):797–821, 1980.

[IY03] Lucian Ilie and Sheng Yu. Follow automata. *Information and Computation*, (186):140–162, 2003.

[Kle65] Stephen Cole Kleene. *Representation of Events in Nerve Nets and Finite Automata*, pages 3–41. Annals of Mathematics Studies. 1965.

[MY60] Robert McNaughton and Hisao Yamada. Regular expressions and state graphs for automata. *IRE Transactions on Electronic Computers*, 9(1):39–47, 1960.

[OF61] Gene Ott and Neil H. Feinstein. Design of sequential machines from their regular expressions. *Journal of the ACM*, 8(4):585–600, 1961.

[SSS88] Seppo Sippu and Eljas Soisalin-Soininen. *Parsing Theory*. EATCS Monographs on Theoretical Computer Science. Springer, 1988.

[Tho68] Ken Thompson. Regular expression search algorithm. *Communications of the ACM*, 11(6):419–422, 1968.

[Wat94] Bruce W. Watson. A taxonomy of finite automata construction algorithms. Technical Report Computing Science Note 93/43, Eindhoven University of Technology, may 1994.

[Woo87] Derick Wood. *Theory of Computation*. John Wiley & Sons, Inc., 1987.

A Appendix

LEMMA 6. $\lhd$ is locally confluent modulo isomorphism.

PROOF. First, assume one of the transitions is labeled by either a product or a sum:

- Let $t_1 = (q, \alpha \bullet \beta, q')$. Upon expansion a bridge-state q'' will be introduced, however the number of arcs leaving and reaching q and q' will remain constant. The structure of A will change insofar as that an arc will be elongated. Since any $\lhd_{t_2}$ will at most have the effect on t_1 that one of its states might be renamed (upon $*1$-expansion), the order of $\lhd_{t_1}, \lhd_{t_2}$ is irrelevant.
- If $t_1{=}(q, \alpha + \beta, q')$, informal reasoning is that an arc is merely doubled. Looking at Def. 4, the booleans $q^+{>}1$ etc. are not changed by such an operation.

Now let both t_i be star-labeled. Note that the statement is trivial, if expansions take place in 'different parts' of the EFA, so let t_1, t_2 share at least a common state. If the transitions are parallel, both will be $*4$-expanded anyway. Further, $*0$-expansion does not change the structure of the state-graph at all, i.e., neither of t_1, t_2 is a loop. So assume $t_1 = (p, \alpha^*, q)$, $t_2 = (q, \alpha^*, r)$ where $p \neq q \neq r$. Some of the possible combinations are shown in Fig. 5, the remaining are a simple exercise.

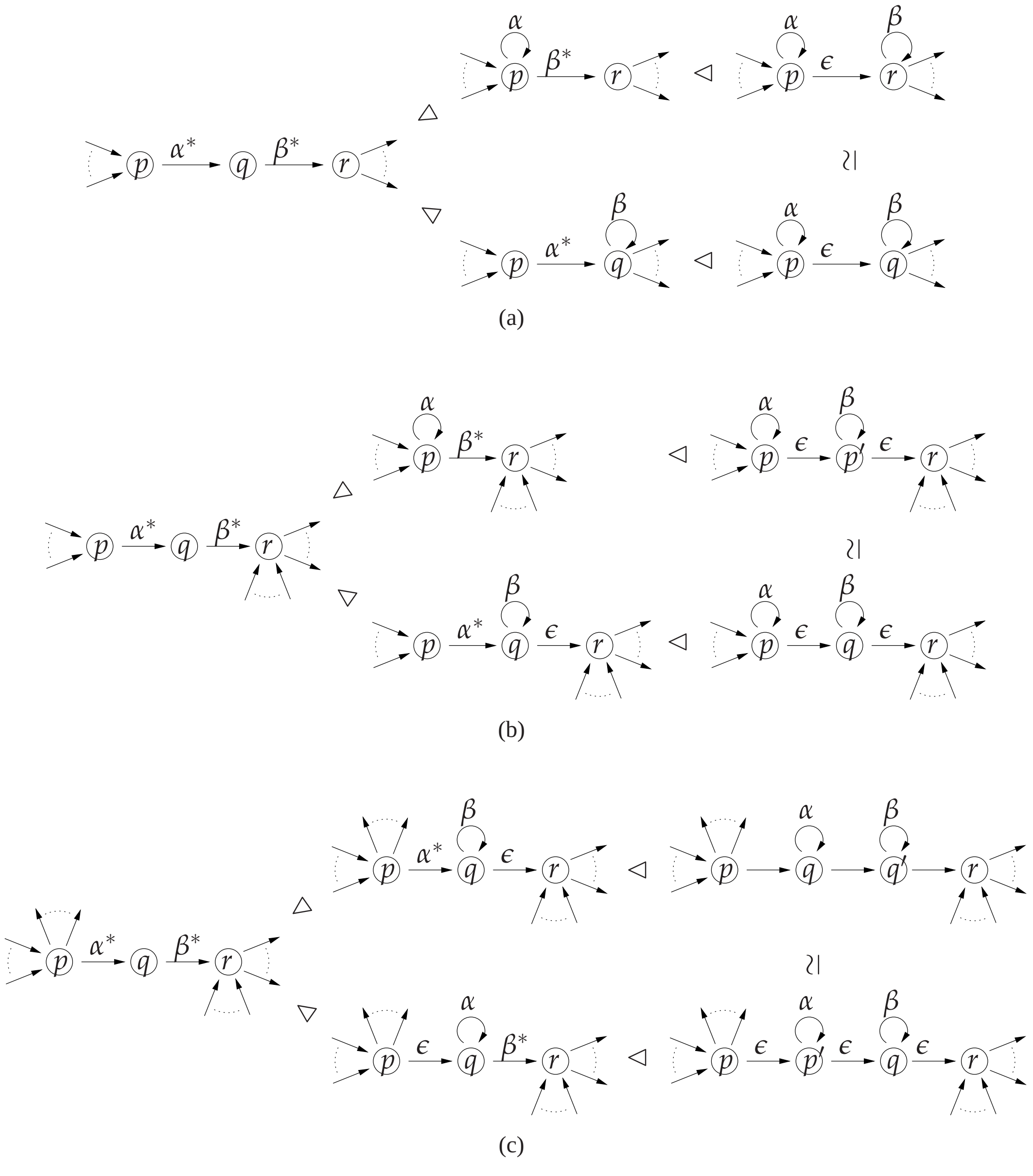


Figure 5: Examples for confluence of expanding consecutive starred transitions. Isomorphism is denoted by $\simeq$.