Simplifying Regular Expressions
A Quantitative Perspective

Hermann Gruber[1] and Stefan Gulan[2]

[1] Institut für Informatik, Universität Gießen,
Arndtstraße 2, D-35392 Gießen, Germany
hermann.gruber@informatik.uni-giessen.de
[2] Fachbereich IV—Informatik, Universität Trier,
Campus II, D-54296 Trier, Germany
gulan@uni-trier.de

Abstract. We consider the efficient simplification of regular expressions and suggest a quantitative comparison of heuristics for simplifying regular expressions. To this end, we propose a new normal form for regular expressions, which outperforms previous heuristics while still being computable in linear time. This allows us to determine an exact bound for the relation between the two prevalent measures for regular expression - size: alphabetic width and reverse polish notation length. In addition, we show that every regular expression of alphabetic width n can be converted into a nondeterministic finite automaton with ε-transitions of size at most $4\frac{2}{5}n+1$, and prove this bound to be optimal. This answers a question posed by Ilie and Yu, who had obtained lower and upper bounds of $4n-1$ and $9n-\frac{1}{2}$, respectively [15]. For reverse polish notation length as input size measure, an optimal bound was recently determined by Gulan and Fernau [14]. We prove that, under mild restrictions, their construction is also optimal when taking alphabetic width as input size measure.

1 Introduction

It is well known that simplifying regular expressions is hard, since alone deciding whether a given regular expression describes the set of all strings, is $\mathbf{PSPACE}$ - complete [17]. As witnessed by a number of recent studies, e.g. [5,10,11,12,13], the descriptional complexity of regular expressions is of great interest, and several heuristics for simplifying regular expressions appear in the literature. These mostly deal with removing only the most obvious redundancies, such as iterated Kleene stars or superfluous occurrences the empty word [15,4,8,9].

We take a quantitative viewpoint to compare such simplifications; namely, we compare the total size of a regular expression (disregarding parentheses) to its alphabetic width. The intuition behind this is as follows: Certain simplifications for regular expressions are of an ad-hoc nature, e.g. the rule $r+r=r$ cannot simplify $a^*+(a+b)^*$. Also, there are rules that are difficult to apply, e.g. if $L(r)\subseteq L(s)$, then $r+s=s$. But there are also simplifications that do not fall in either category, such as the reduction rules suggested in [15,4,8,9,16]. In this paper, we suggest a *strong star normal form* of regular expressions, which is a variation of the star normal form defined in [4]. This

A.-H. Dediu, H. Fernau, and C. Martín-Vide (Eds.): LATA 2010, LNCS 6031, pp. 285–296, 2010.

normal form achieves an optimal ratio when comparing expression size to alphabetic width, and can be computed as efficiently as the original star normal form.

For converting regular expressions into small ε-NFAs, an optimal construction was found recently in [14]. Here, *optimal* means that the algorithm attains the best possible ratio of *expression size* to *automaton size*. Ilie and Yu [15] asked for the optimal quotient if expression size is replaced with *alphabetic width*; they obtained an upper bound of roughly 9. We resolve this open problem by showing that the quotient equals $4\frac{2}{5}$. In fact, we prove that the construction from [14] attains this bound if the input expression is in strong star normal form. We move on to show that this still holds, under very mild restrictions, also for expressions not in star normal form. Our results suggest that this construction of ε-NFAs from regular expressions is optimal in a robust sense.

2 Basic Notions

Let Σ be a set of symbols, called *letters*. Regular expression over Σ, or just *expressions*, are defined as follows: Every letter is an expression and if r_1 and r_2 are expressions, so are (r_1+r_2), $(r_1\cdot r_2)$, $(r_1)^?$ and $(r_1)^*$. The language denoted by an expression r, written $L(r)$, is defined inductively: $L(a) = \{a\}$, $L(r_1 + r_2) = L(r_1) \cup L(r_2)$, $L(r_1 \cdot r_2) = L(r_1) \cdot L(r_2)$, $L(r_1^?) = \{\varepsilon\} \cup L(r_1)$ and $L(r_1^*) = L(r_1)^*$. A language is called regular if it is definable by an expression.

We deviate from the convention by omitting symbols denoting the empty set and the empty word, while allowing for a special operator that adds the empty word to a language. The disadvantages of our definition are minor—we cannot describe the degenerate languages $\emptyset$ and $\{\varepsilon\}$; on the plus side, our syntax prevents *a priori* the construction of many kinds of unnatural and redundant expressions, such as $\varepsilon \cdot r$ or $\emptyset^*$.

There are two prevalent measures for the length of expressions: The *alphabetic width* of r, denoted $\mathrm{alph}(r)$ is defined as the total number of occurrences of letters in r. The second measure is the reverse polish notation length. To allow for comparison with related works, e.g., [8,15], we define the (abbreviated) reverse polish notation length of r as $\mathrm{arpn}(r) = |r|_\Sigma + |r|_+ + |r|_\cdot + |r|_* + |r|_?$, and its unabbreviated rpn-length as $\mathrm{rpn}(r) = \mathrm{arpn}(r) + |r|_?$. This reflects the fact that replacing each subexpression of the form $s^?$ with $s + \varepsilon$ increases the overall length by 1 each time. The alphabetic width of a regular language L is defined as the minimum alphabetic width among all expressions denoting L, and is denoted $\mathrm{alph}(L)$. The notions $\mathrm{rpn}(L)$ and $\mathrm{arpn}(L)$ are defined correspondingly.

Some notions from term rewriting are needed:Let S be a set, and let $\to$ be a relation on S. Let $\to^*$ denote the transitive closure of $\to$. Two elements $b, c \in S$ are called *joinable*, if some $d \in S$ satisfies $b \to^* d$ and $c \to d$. The relation $\to$ is *confluent*, if for all $a, b, c \in S$ with $a \to^* b$ and $a \to^* c$, the elements b and c are joinable. It is *locally confluent*, if for all $a, b, c \in S$ with $a \to b$ and $a \to c$, the elements b and c are joinable. The relation is *terminating*, if there is no infinite descending chain $a_1 \to a_2 \to \cdots$.

It is easily proven that if $\to$ is confluent and terminating, then each element has a unique normal form, see e.g. [2, Thm. 2.1.9]. Indeed for unique normal forms, we only need to establish local confluence instead of confluence: Newman's Lemma states that if a terminating relation is locally confluent, then it is confluent ([18], see also [2, Lem. 2.7.2]).

3 Alphabetic Width versus Reverse Polish Notation Length

We adapt the *star normal form* of expressions, proposed by Brueggemann-Klein [4], to our needs.

Definition 1. *The operators* $\circ$ *and* $\bullet$ *are defined on expressions as follows: For the first operator, let* $a^\circ = a$*, for* $a \in \Sigma$*,* $(r+s)^\circ = r^\circ + s^\circ$*,* $r^{?\circ} = r^\circ$*,* $r^{*\circ} = r^\circ$*, and*

$$(rs)^\circ = \begin{cases} rs, & \text{if } \varepsilon \notin L(rs) \\ r^\circ + s^\circ & \text{else} \end{cases} .$$

The second operator is given by: $a^\bullet = a$*, for* $a \in \Sigma$*,* $(r+s)^\bullet = r^\bullet + s^\bullet$*,* $(rs)^\bullet = r^\bullet s^\bullet$*,* $r^{*\bullet} = r^{\bullet\circ *}$*, and*

$$r^{?\bullet} = \begin{cases} r^\bullet & \text{, if } \varepsilon \in L(r) \\ r^{\bullet ?} & \text{otherwise} \end{cases} .$$

The strong star normal form *of an expression* r *is then defined as* $r^\bullet$*.*

Observe that, e.g., the expression $(\emptyset + a)^* + \varepsilon \cdot b + \emptyset \cdot c \cdot (d + \varepsilon + \varepsilon)$ in unabbreviated syntax is in star normal form, so the relative advantage of strong star normal form should be obvious. The difference to star normal form merely consists in using abbreviated syntax and in the addition of a rule for computing $r^{?\bullet}$. All the statements in the original work [4, Thm. 3.1, Lem. 3.5, 3.6, 3.7] regarding $^\circ$ and $^\bullet$ carry over to our variation.

We compare rpn-length and alphabetic width of expressions in strong star normal form. To this end, for an expression r in abbreviated syntax, define $\omega(r) = |r|_? + |r|_*$, that is, ω counts the total number of occurrences of unary operators in r. The following property is evident from the definition of $\bullet$ and $\circ$; a similar statement concerning rpn-length is found in [4].

Lemma 1. *Let r be an expression. Then* $\omega(r^\bullet), \omega(r^\circ) \leq \omega(r)$*, and* $\omega(r^{*\circ}) \leq \omega(r^*) - 1$*.*

Lemma 2. *Let* r *be an expression, then* $\omega(r^\bullet) \leq \operatorname{alph}(r^\bullet)$*, if* $\varepsilon \in L(s)$*, and* $\omega(r^\bullet) \leq \operatorname{alph}(r^\bullet) - 1$ *otherwise.*

Proof. By lexicographic induction on the pair (n, h), where $n = \operatorname{alph} r^\bullet$, and h is the height of the parse of r. The base case is $(1, 1)$, i.e., $r^\bullet \in \Sigma$, for which the statement clearly holds. Assume the claim is true for expressions of alphabetic width at most $n-1$ and for expressions of alphabetic width n and height at most $k-1$. The nontrivial cases for the induction step are $r = s^?$ and $r = s^*$. In the first case, we have $r^\bullet = s^\bullet$, if $\varepsilon \in L(s)$. Applying the induction hypothesis to $s^\bullet$ yields

$$\operatorname{alph}(r^\bullet) = \operatorname{alph}(s^\bullet) \geq \omega(s^\bullet) = \omega(r^\bullet).$$

If $\varepsilon \notin L(s)$, then $r^\bullet = s^{\bullet ?}$, where again the induction hypothesis applies for s. This time, we obtain

$$\operatorname{alph}(r^\bullet) = \operatorname{alph}(s^\bullet) \geq \omega(s^\bullet) + 1 = \omega(r^\bullet).$$

In case $r = s^*$, we need to distinguish by the structure of r. The easy cases are $r = s^{?*}$ and $r = s^{**}$: here, $r^\bullet = s^{*\bullet}$ and the claim holds by induction. If $r = (s+t)^*$, expansion of the definition gives

$$r^\bullet = (s^{*\bullet\circ} + t^{*\bullet\circ})^*.$$

Since both $s^{*\bullet}$ and $t^{*\bullet}$ must have alphabetic width strictly less than n, and since both describe the empty word, we apply the inductive hypothesis to obtain

$$\operatorname{alph}(r^\bullet) = \operatorname{alph}(s^{*\bullet}) + \operatorname{alph}(t^{*\bullet}) \geq \omega(s^{*\bullet}) + \omega(t^{*\bullet}).$$

Now $\omega(s^{*\bullet\circ}) \leq \omega(s^{*\bullet}) - 1$, and similar for $t^{*\bullet\circ}$, we deduce that $\omega(r^\bullet) \leq \operatorname{alph}(r^\bullet) - 2$, which completes the induction step for this case.

For the case where $r = (st)^*$, we have $r^\bullet = (s^\bullet t^\bullet)^{\circ *}$ and the induction goes through if at least one of s and t does not describe the empty word. If however $\varepsilon \in L(s) \cap L(t)$, then it is easy to prove under this condition that $r^\bullet = (s+t)^{*\bullet}$, a case we already dealt with a few lines above in this proof. □

Theorem 1. *Any regular language L satisfies* $\operatorname{arpn}(L) \leq 3\operatorname{alph}(L) - 1$ *and* $\operatorname{rpn}(L) \leq 4\operatorname{alph}(L) - 1$.

Proof. Let r be an expression, in abbreviated syntax, of minimum alphabetic width denoting L. Then the parse tree of $r^\bullet$ has $\operatorname{alph}(r)$ many leaves. Disregarding unary operators, this is a binary tree with $\operatorname{alph}(r) - 1$ internal vertices that correspond to occurrences of binary operators in r. Since there are at most $\operatorname{alph}(r)$ many occurrences of unary operators, we have $\operatorname{arpn}(r^\bullet) \leq 3\operatorname{alph}(L) - 1$ and $\operatorname{rpn}(r^\bullet) \leq \operatorname{arpn}(r^\bullet) + \omega(r^\bullet) \leq 4\operatorname{alph}(L) - 1$. □

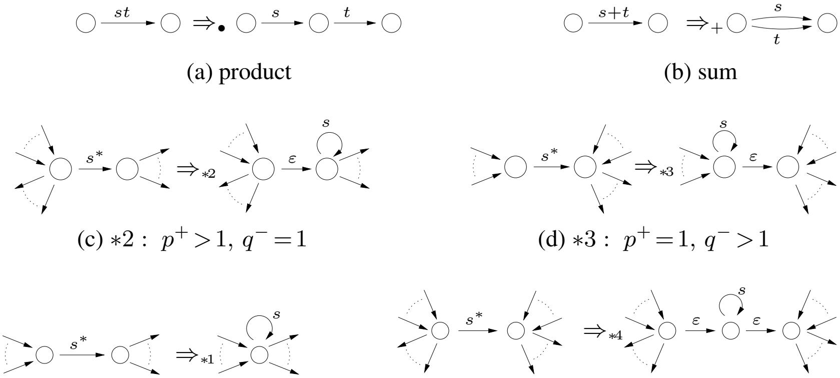
Thus size and alphabetic width can differ at most by a factor of 4 in unabbreviated syntax. Previous bounds, which were based on other simplification paradigms, by Ilie and Yu [15] and by Ellul et al. [8] only achieved factors of 6 and 7, respectively, in place of 4. For abbreviated syntax, we will later show that the bound of the form $3n - 1$ is best possible. Also note that strong star normal form subsumes all of the previous simplification heuristics from [4,8,15].

4 Constructing ε-NFAs from Regular Expressions, Revisited

We show that under mild restrictions, the construction given by Gulan and Fernau [14] subsumes the conversion of the input expression into strong star normal form. This construction is essentially a replacement system on digraphs, that are arc-labeled by regular expressions or the symbol ε. Such objects are called *extended finite automata* (EFAs), as they generalize (conventional) finite automata; consult Wood [20] for a proper introduction. The replacements are called *conversions*; they come in two flavors:

- A transition labeled by a regular expression may be replaced wrt. the labels root. These conversions, called *expansions*, are depicted in Fig. 1.
- A substructure defined by ε-transitions may be replaced by a smaller equivalent. These conversions are also called *eliminations*, they are shown in Fig. 2.

Since ε-transitions are allowed in EFAs, we treat $r^?$ implicitly as $r + \varepsilon$. We call the lhs of i-expansion or i-elimination an *i-anchor*, and write $E \Rightarrow_i E'$ if E' is derived from replacing an i-anchor in E with its according rhs. If the type of conversion is irrelevant, we simply write $E \Rightarrow E'$, and denote a (possibly empty) series of conversions from



(a) product

(b) sum

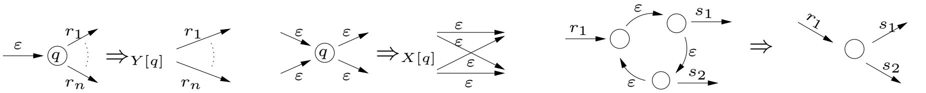
(c) $*2 :\ p^{+} > 1,\ q^{-} = 1$

(d) $*3 :\ p^{+} = 1,\ q^{-} > 1$

(e) $*1 :\ p^{+} = 1,\ q^{-} = 1$; merge p and q

(f) $*4 :\ p^{+} > 1,\ q^{-} > 1$; introduce a new state

Fig. 1. Expanding transitions (p, r, q) for nontrivial r. If $r = s^*$, the out-degree p^+ of p and the in-degree q^- of q need to be considered.



(a) Y-elimination, requires $q^- = 1$

(b) X-elimination, requires $q^- = q^+ = 2$

(c) O-elimination

Fig. 2. Eliminating substructures with ε-labeled transitions. Reverting all transitions in (a), and demanding that $q^+ = 1$ yields a further Y-rule.

E to E' with $E \Rightarrow^* E'$. An expression r over Σ is identified with the trivial EFA $A_r^0 := (\{q_0, q_f\}, \Sigma, \{(q_0, r, q_f)\}, q_0, q_f)$. On input r, the construction is initialized with A_r^0, which is successively and exhaustively converted to an ε-NFA, denoted A_r. We slightly restrict the applicability of conversions by two rules:

(R1) As long as any conversion other than $\Rightarrow_X$ is possible, X-elimination must not be applied.

(R2) If two X-anchors share ε-transitions the one from which they are leaving is to be eliminated.

Other than that, conversions may be applied in any order. Note that (R2) is sound: cyclic elimination preference among X-anchors would imply the existence of an O-anchor, which, due to (R1), would be eliminated first. The conversion process is split into a sequence of conversions without X-eliminations, followed by one with X- eliminations only. This is due to

Proposition 1. *Let $E \Rightarrow_X E'$ respect (R1). Then E' contains only X-anchors, if any.*

Proof. Since $E \Rightarrow_X E'$ respects (R1), E contains only X-anchors. Neither complex labels nor cycles, particularly no O-anchors, are introduced upon X-elimination.

Assume $E \Rightarrow_{X[q]} E' \Rightarrow_{Y[p]} E''$ is a valid conversion sequence, then p and q are adjacent in E, since the Y-anchor in E' results from the preceding X-elimination. Let (p, ε, q) be the transition connecting p and q in E, then in E', $p^+ = 2$, hence $p^- = 1$. But the in-degree of p is not changed by this X-elimination, so $p^- = 1$ in E, too. But then, E contains an Y-anchor centered in p, contradicting the assumption that the conversion respects (R1).

To designate the transition between the two phases, let A_r^k be the first EFA in the sequence $A_r^0 \Rightarrow A_r^1 \Rightarrow \cdots \Rightarrow A_r$ that allows for no conversion besides possibly X-elimination; we denote this automaton X_r. If X-elimination does not occur at all upon full conversion, then $X_r = A_r$.

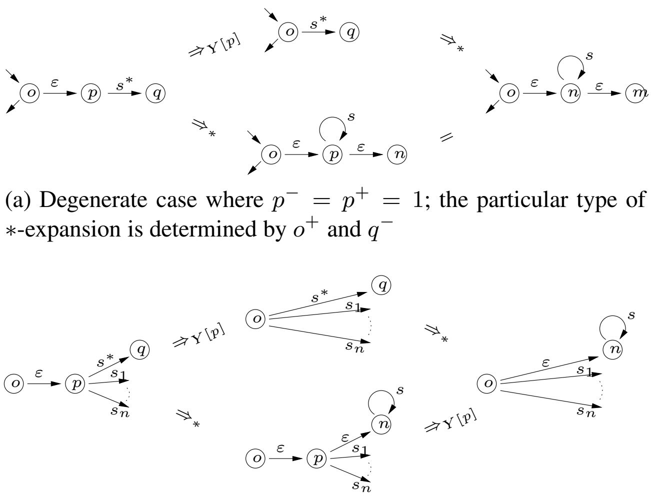
We show that the conversions other than X-elimination are locally confluent. To this end, we write $E_1 \cong E_2$, if E_1 and E_2 are joinable. Since no infinite conversion sequences are possible, Newman's Lemma implies that X_r is unique.

Theorem 2. *The replacement-system consisting of* $\Rightarrow_+$, $\Rightarrow_\bullet$, $\Rightarrow_{*i}$, $\Rightarrow_Y$ *and* $\Rightarrow_O$ *is locally confluent on the class of EFAs.*

Proof. We claim that $E \Rightarrow_i E_1$ and $E \Rightarrow_j E_2$ for $i, j \in \{+, \bullet, *1, *2, *3, *4, Y, O\}$, implies $E_1 \cong E_2$. This is trivial if the conversions occur in disjoint subautomata, so assume the relevant anchors share at least a state. We assume that at least one of i, j is Y or O, the remaining cases are discussed in [14, Lem. 6]. We distinguish by i:

- $i = Y[p]$: Let (o, ε, p) be the ε-transition to be removed, and assume $\Rightarrow_j$ is an expansion, then one of the labels r_k as in Fig. 2(a) is a product, a sum or a starred expression. If r_k is a sum or a product, it is easy to see that the order of $\Rightarrow_i$ and $\Rightarrow_j$ is interchangeable. We sketch the cases involving $*$-expansion in Fig. 3. The three cases arising when both conversions are Y-eliminations, are illustrated in Fig. 4.
- $i = O$: O-elimination comes down to removing the ε-transitions forming a cycle, followed by merging the cycle-states into a selected one among them, call this the *merge-state*. If $\Rightarrow_j$ is the expansion of $t = (p, s, q)$, assume p lies on the cycle, while q does not. Choose p as the merge-state, then t remains unaffected from O-elimination, hence expansion introduces the same elements before and after O-elimination. If q is part of the cycle but not p, or $p = q$, choose q as the merge-state. If both p and q lie on the cycle and $p \neq q$, the case of $j = *4$ is detailed in Fig. 5, the remaining cases where j is an expansion are easily dealt with in the same way. Next consider the case that $\Rightarrow_j$ is $Y[q]$-elimination, for some state q, and where q is part of the ε-cycle relevant for O-elimination—the case where q is not on the ε-cycle in question would be again easy. By definition of Y-elimination, we must have $q^- = 1$ (resp. $q^+ = 1$ in the case of reverse Y-elimination), and there must be exactly one ε-transition entering (resp. leaving) the state q. Since $q^- = 1$ (resp. $q^+ = 1$), this transition is necessarily part of the ε-cycle in question. Hence, if O-elimination is applied first, it subsumes Y-elimination; otherwise, Y-elimination may be considered as the first merging step of O-elimination, followed by merging a smaller cycle.

  Finally, if $\Rightarrow_j$ also denotes O-elimination, there is at least one common state c to both cycles, which we chose as the merge-state. Regardless of the order, both cycles may be merged into c, thus yielding the same EFA. □



(a) Degenerate case where $p^- = p^+ = 1$; the particular type of $*$-expansion is determined by o^+ and q^-

(b) General case

Fig. 3. Local confluence of cases involving Y-elimination and $*$-expansion. The state denoted n is either q or a newly introduced state, according to q^-. Note that reverting all transitions in the figures yields further valid cases.

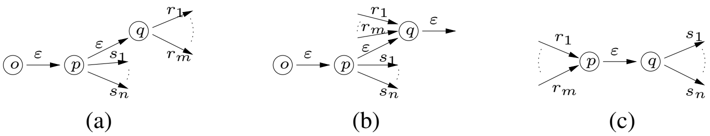


Fig. 4. Elimination-conflicts between overlapping Y-anchors centered in p and q. In (a) and (b), the resulting EFA is invariant under the order of removal. In (c) only one anchor may be eliminated, however, the resulting EFAs are isomorphic.

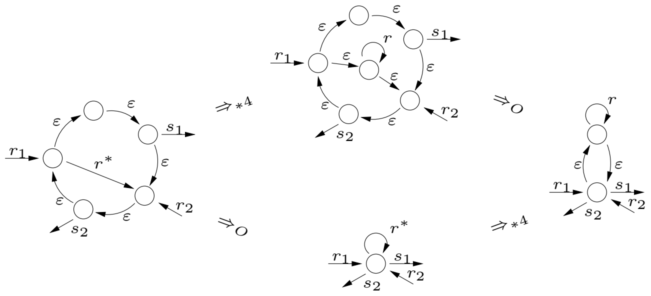


Fig. 5. Conflict between cycle-elimination and expanding a transition connecting two distinct states of the cycle

We omit proving that the conversion from X_r to A_r is also locally confluent, which is due to restriction (R2). This implies that A_r is unique, too.

We add an almost trivial linear-time preprocessing step on the input expression, called *mild simplification*: Every occurrence of $s^?$ in r, s.t. $\varepsilon \in L(s)$ is replaced with s. The expression such built from r is denoted $\mathrm{simp}(r)$, it can be computed in linear time on the parse of r in a bottom-up manner. Without proof, we mention that computing the strong star normal form subsumes mild simplification:

Lemma 3. *Let r be a regular expression, then* $\mathrm{simp}(r)^\bullet = \mathrm{simp}(r^\bullet) = r^\bullet$

On input r, we mildly simplify it first and then compute $A^0_{\mathrm{simp}(r)}$. The size $|A|$ of an EFA A is defined as the number of its states and transitions *combined*. Mild simplification is a reasonable first step in order to get smaller ε-NFAs:

Lemma 4. *For any expression r, $|A_{\mathrm{simp}(r)}| \leq |A_r|$*

Proof. Let E_1 be an EFA with transition $t = (p, s^?, q)$, and let E_2 be the EFA obtained from E_1 by replacing t with (p, s, q). Expanding t in E_1 yields E_1'; the difference between E_1' and E_2 is an additional transition (p, ε, q) in E_1'. Now p^+ and q^- are bigger in E_1' than in E_2 — if $s = t^*$, expanding (p, s, q) in E_1' introduces at least as many elements in E_2. On the other hand, removal of p or q in E_1' may result from X- or cycle-elimination, then however, Y- or cycle-elimination would be applicable in E_2. In short, converting E_1' does not yield an ε-NFA which is smaller than the one reached by converting E_2. Since mildly simplifying an expression boils down to replacing some occurrences of $s^?$ with s (in labels), the statement follows. □

The remaining part of this section deals with invariant cases of the construction under $^\circ$ and $^\bullet$. To this end, for a transition $t = (p, r, q)$ let $t^\circ := (p, r^\circ, q)$ and $t^\bullet := (p, r^\bullet, q)$. Note that since the conversions are locally confluent when respecting (R1) and (R2), $\cong$ is an equivalence relation on the class of EFAs.

Lemma 5. *Let E_1 be an EFA with looping transition $l = (q, r, q)$, and let E_2 be the EFA obtained from E_1 by replacing l with l°. Then $E_1 \cong E_2$.*

Proof. If $r \in \Sigma$ then $r = r^\circ$, satisfying the claim. Let E_1 and E_2 be as above and assume the claim is true for loops labeled s or t. Let r be

- $s + t$: l is replaced by $(q, s, q), (q, t, q)$, while l° is replaced by $(q, s^\circ, q), (q, t^\circ, q)$. By assumption, the pairs are interchangeable, hence so are l and l°
- $s^?$: l is replaced by loops $(q, \varepsilon, q), (q, s, q)$, the first of which is an ε-cycle, hence eliminated, while the second may by assumption be replaced with $(q, s^\circ, q) = (q, s^{?\circ}, q) = l^\circ$.
- s^*: $*4$-expansion is applied, introducing an ε-cycle $\{(q, \varepsilon, q'), (q', \varepsilon, q)\}$ and a loop (q', s, q'). Eliminating the cycle identifies q and q', yielding (q, s, q) which may by assumption be replaced with $(q, s^\circ, q) = l^\circ$
- st: If $\varepsilon \notin L(st)$, then $(st)^\circ = st$ and nothing needs to be proven. So assume $\varepsilon \in L(st)$, implying $\varepsilon \in L(s)$ and $\varepsilon \in L(t)$. Let E_1' be the EFA after fully expanding r, without intermediate elimination steps. The first expansion-step replaces t_l with $\{(q, s, q'), (q', t, q)\}$ — both q and q' are still present in E_1', where they lie on an

ε-cycle. Consider cycle-elimination in 'slow-motion': in a first step, only q and q' are merged, resulting in a volatile intermediate which happens to be isomorphic to the EFA constructed from fully expanding $l^\circ = (q, s^\circ + t^\circ, q)$ in E_2. A second step merges the remaining states, which is equivalent to two cycle-eliminations. □

A more general result can be established for mildly simplified expressions:

Lemma 6. *Let $A_r^0 \Rightarrow^* E_1$ for mildly simplified r. Let $t = (p, r, q)$ be any transition in E_1, and let E_2 be as E_1 except that t is replaced with $t^\bullet$. Then $E_1 \cong E_2$.*

Proof. The statement is true for letters. Assume it is true for labels s and t, and let E_1 and E_2 be as above. Let r be

- $s^?$: expansion replaces t with $\{(p, s, q), (p, \varepsilon, q)\}$, the first of which may by assumption be replaced with $(p, s^\bullet, q)$. Since r is mildly simplified, $\varepsilon \notin L(s)$ therefore $r^\bullet = s^{?\bullet} = s^{\bullet ?}$; this implies that $(p, r^\bullet, q)$ is expanded into $(p, s^\bullet, q)$ and (p, ε, q) as well.
- s^*: expanding t yields a looping transition $l = (p', s, p')$, which may by assumption be replaced with $l^\bullet$ and by Lemma 5 with $l^{\bullet\circ}$. Clearly, expanding $t^\bullet = (q, s^{\bullet\circ *}, q')$ results in $l^{\bullet\circ}$, too.

The remaining cases are straightforward. □

Theorem 3. *Let r be mildly simplified, then the ε-*NFA *constructed from r is isomorphic to the one constructed from its strong star normal form, that is, $A_r \cong A_{r^\bullet}$.*

Proof. Lemma 6 implies $A_r^0 \cong A_{r^\bullet}^0$. □

Together with Lemma 3, this shows that the construction is invariant under taking strong star normal form. Differently put, strong star normal form is implicitly computed upon conversion of mildly simplified regular expressions.

5 Alphabetic Width and the Size of ε-NFAs

Let the *size* of an ε-NFA be its number of states plus its number of transitions. The following question regarding the size of ε-NFAs was posed by Ilie and Yu.

Problem 1. Given a regular expression of alphabetic width n, what is the optimal bound on the size of an equivalent ε-NFA in terms of n?

Ilie and Yu provide a bound of $9n - \frac{1}{2}$; they remark that this does not appear to be close to optimal. The construction we discussed in the previous section was shown to give following bound in terms of rpn-length on the size of the constructed ε-NFA:

Theorem 4 ([14]). *Let r be a regular expression of unabbreviated rpn-length n. Then the constructed ε-*NFA *A_r has size at most* $22/15(n+1)+1$. *There are infinitely many regular languages for which this bound is tight.*

The original work does not consider abbreviated syntax for regular expressions. Fortunately, subexpressions of the form $r + \varepsilon$ do not contribute to the hardness of the conversion problem. The following bound in terms of *abbreviated* rpn-length is slightly stronger.

Theorem 5. *Let r be an expression of abbreviated rpn-length n. Then the constructed* ε-NFA A_r *has size at most* $22/15(n+1)+1$. *There are infinitely many regular languages for which this bound is tight.*

Proof. The analysis is the same as given in [14], except for obvious modifications to the proof of [14, Thm. 10], which is the only place where we take the use of abbreviated syntax into account. The fact that this bound is tight for infinitely many regular languages trivially carries over. □

Together with Thms. 1 and 3, we obtain the following upper bound:

Theorem 6. *Let r be a regular expression of alphabetic width n. If r is mildly simplified, then the constructed* ε-NFA A_r *has size at most* $4\frac{2}{5}n+1$. *There are infinitely many regular languages for which this bound is tight.*

Proof. Let r be mildly simplified with $\operatorname{alph}(r) = n$. Then Thm. 3 implies that A_r is identical to $A_{r^\bullet}$ and we know from Thm. 1 that $\operatorname{arpn}(r^\bullet) \leq 3n-1$. Plugging this into the statement of Thm. 5, it follows that the ε-NFA $A_{r^\bullet}$, constructed from $r^\bullet$, has size at most $22/15(3n-1+1)+1 = 4\frac{2}{5}n+1$.

Gulan and Fernau [14] also give an infinite family of regular expressions r_n showing that the bound $22/15(m-1)+1$ on the size of an ε-NFA equivalent to a regular expression of rpn-length m is optimal: For $k \geq 1$, they define the regular expression

$$r_k = \prod_{i=1}^{k} (a_i^* + b_i^*) \cdot (c_i^* + d_i^* + e_i^*)$$

of rpn-length $m = 15k-1$ and prove that every equivalent ε-NFA has size at least $22k+1 = 22/15(m+1)+1$. Since the alphabetic width of r_k is $\ell = 5k$, this shows that the bound of $22k+1 = 4\frac{2}{5}\ell+1$ stated in the theorem is tight for infinitely many regular languages. □

The examples from the last proof can be used to prove that the bound from Thm. 1 is tight in the abbreviated case:

Theorem 7. *There is an infinite family L_n of regular languages such that* $\operatorname{alph}(L_n) \leq n$, *whereas* $\operatorname{arpn}(L_n) \geq 3n-1$.

Proof. Consider the language L_n described by the expression

$$r_k = \prod_{i=1}^{k} (a_i^* + b_i^*)(c_i^* + d_i^* + e_i^*).$$

For $n = 5k$ and $L_n = L(r_{5k})$, we have $\operatorname{alph}(L_n) = 5k = n$. But the existence of an expression of abbreviated rpn-length less than $3n-1 = 15k-1$ would imply with Theorem 5 that there exists an ε-NFA of size less than $22k+1$ accepting L_n, which contradicts Thm. 4. □

6 Conclusion and Further Research

As equivalence of expressions is **PSPACE**-complete [17] and not finitely axiomatizable [1,7], a normal form that assigns a unique expression to each regular language, might be difficult to obtain. Ideally, we would like a normal form that realizes minimum alphabetic width and minimum rpn-length, and that is efficiently computable — two criteria, that would apparently contradict the above negative theoretical results.

We have suggested a robust notion of reduced expressions, the strong star normal form. This notion satisfies at least the latter two criteria, in the sense that each regular language, admits at least one regular expression in star normal form of minimum rpn-length and of minimum alphabetic width, while being computable in linear time. Our notion subsumes previous attempts at defining such a notion [4,8,15].

Furthermore, we showed that the strong star normal form proves useful in various contexts: Apart from a prior application in the context of the construction of ε-free NFAs [6], we gave two further applications.

The first concerns the relation between different complexity measures for regular expressions, namely alphabetic width and (abbreviated) rpn-length. With the aid of strong star normal form, we were able to determine the optimal bound, witnessing superiority of this concept over previous attempts at defining such a notion of irreducibility, which yield only loose bounds [8,15].

The second application concerns the comparison of descriptional complexity measures across different representations, namely alphabetic width on the one hand, and the minimum size of equivalent ε-NFAs on the other hand. Here, we applied a construction proposed recently by Gulan and Fernau [14]: Under a mild additional assumption, this construction already incorporates all simplifications offered by strong star normal form. While this alone adds to the impression of robustness of the construction, we also proved an optimal bound on the relation between alphabetic width and the size of finite automata, and we showed that this bound is attained by the mentioned construction.

We believe that there are various further applications outside the theoretical domain. For instance, the fastest known algorithm [3] for regular expression matching is still based on the classical construction due to Thompson [19]. While better constructions for ε-NFAs may not improve the asymptotic worst-case running time, we hope that these can still lead to noticeably better practical performance of NFA-based regular expression engines.

References

1. Aceto, L., Fokkink, W., Ingólfsdóttir, A.: On a question of A. Salomaa: the equational theory of regular expressions over a singleton alphabet is not finitely axiomatizable. Theoretical Computer Science 209(1), 163–178 (1998)
2. Baader, F., Nipkow, T.: Term Rewriting and All That. Cambridge University Press, Cambridge (1998)
3. Bille, P., Thorup, M.: Faster regular expression matching. In: ICALP 2009. LNCS, vol. 5555, pp. 171–182. Springer, Heidelberg (2009)
4. Brüggemann-Klein, A.: Regular expressions into finite automata. Theoretical Computer Science 120(2), 197–213 (1993)

5. Caron, P., Champarnaud, J.M., Mignot, L.: Multi-tilde operators and their Glushkov automata. In: Dediu, A.H., Ionescu, A.M., Martín-Vide, C. (eds.) LATA 2009. LNCS, vol. 5457, pp. 290–301. Springer, Heidelberg (2009)
6. Champarnaud, J.M., Ouardi, F., Ziadi, D.: Normalized expressions and finite automata. International Journal of Algebra and Computation 17(1), 141–154 (2007)
7. Conway, J.H.: Regular Algebra and Finite Machines. Chapman and Hall, Boca Raton (1971)
8. Ellul, K., Krawetz, B., Shallit, J., Wang, M.: Regular expressions: New results and open problems. Journal of Automata, Languages and Combinatorics 10(4), 407–437 (2005)
9. Frishert, M., Cleophas, L.G., Watson, B.W.: The effect of rewriting regular expression on their accepting automata. In: Ibarra, O.H., Dang, Z. (eds.) CIAA 2003. LNCS, vol. 2759, pp. 304–305. Springer, Heidelberg (2003)
10. Gelade, W., Martens, W., Neven, F.: Optimizing schema languages for XML: Numerical constraints and interleaving. SIAM Journal on Computing 38(5), 2021–2043 (2009)
11. Gelade, W., Neven, F.: Succinctness of the complement and intersection of regular expressions. In: Symposium on Theoretical Aspects of Computer Science. Number 08001 in Dagstuhl Seminar Proceedings, pp. 325–336 (2008)
12. Gruber, H., Holzer, M.: Finite automata, digraph connectivity, and regular expression size. In: Aceto, L., Damgård, I., Goldberg, L.A., Halldórsson, M.M., Ingólfsdóttir, A., Walukiewicz, I. (eds.) ICALP 2008, Part II. LNCS, vol. 5126, pp. 39–50. Springer, Heidelberg (2008)
13. Gruber, H., Johannsen, J.: Optimal lower bounds on regular expression size using communication complexity. In: Amadio, R.M. (ed.) FOSSACS 2008. LNCS, vol. 4962, pp. 273–286. Springer, Heidelberg (2008)
14. Gulan, S., Fernau, H.: An optimal construction of finite automata from regular expressions. In: FSTTCS 2008. Number 08004 in Dagstuhl Seminar Proceedings, pp. 211–222 (2008)
15. Ilie, L., Yu, S.: Follow automata. Information and Computation 186(1), 140–162 (2003)
16. Lee, J., Shallit, J.: Enumerating regular expressions and their languages. In: Domaratzki, M., Okhotin, A., Salomaa, K., Yu, S. (eds.) CIAA 2004. LNCS, vol. 3317, pp. 2–22. Springer, Heidelberg (2005)
17. Meyer, A.R., Stockmeyer, L.J.: The equivalence problem for regular expressions with squaring requires exponential space. In: FOCS 1972, pp. 125–129. IEEE Computer Society, Los Alamitos (1972)
18. Newman, M.: On theories with a combinatorial definition of “equivalence”. Annals of Mathematics 43(2), 223–243 (1942)
19. Thompson, K.: Regular expression search algorithm. Communications of the ACM 11(6), 419–422 (1968)
20. Wood, D.: Theory of Computation. John Wiley & Sons, Inc., Chichester (1987)