Succinctness of the Complement and Intersection of Regular Expressions
Wouter Gelade, Frank Neven

To cite this version:

HAL Id: hal-00226864
https://hal.archives-ouvertes.fr/hal-00226864
Submitted on 30 Jan 2008

HAL is a multi-disciplinary open access archive for the deposit and dissemination of scientific research documents, whether they are published or not. The documents may come from teaching and research institutions in France or abroad, or from public or private research centers.

L’archive ouverte pluridisciplinaire HAL, est destinée au dépôt et à la diffusion de documents scientifiques de niveau recherche, publiés ou non, émanant des établissements d’enseignement et de recherche français ou étrangers, des laboratoires publics ou privés.
SUCCINCTNESS OF THE COMPLEMENT AND INTERSECTION OF REGULAR EXPRESSIONS

WOUTER GELADE AND FRANK NEVEN

Hasselt University and Transnational University of Limburg, School for Information Technology
E-mail address: firstname.lastname@uhasselt.be

ABSTRACT. We study the succinctness of the complement and intersection of regular expressions. In particular, we show that when constructing a regular expression defining the complement of a given regular expression, a double exponential size increase cannot be avoided. Similarly, when constructing a regular expression defining the intersection of a fixed and an arbitrary number of regular expressions, an exponential and double exponential size increase, respectively, can in worst-case not be avoided. All mentioned lower bounds improve the existing ones by one exponential and are tight in the sense that the target expression can be constructed in the corresponding time class, i.e., exponential or double exponential time. As a by-product, we generalize a theorem by Ehrenfeucht and Zeiger stating that there is a class of DFAs which are exponentially more succinct than regular expressions, to a fixed four-letter alphabet. When the given regular expressions are one-unambiguous, as for instance required by the XML Schema specification, the complement can be computed in polynomial time whereas the bounds concerning intersection continue to hold. For the subclass of single-occurrence regular expressions, we prove a tight exponential lower bound for intersection.

1. Introduction

The two central questions addressed in this paper are the following. Given regular expressions $r, r_1, \ldots, r_k$ over an alphabet $\Sigma$,

1. what is the complexity of constructing a regular expression $r_\neg$ defining $\Sigma^* \setminus L(r)$, that is, the complement of $r$?
2. what is the complexity of constructing a regular expression $r_\cap$ defining $L(r_1) \cap \cdots \cap L(r_k)$?

In both cases, the naive algorithm takes time double exponential in the size of the input. Indeed, for the complement, transform $r$ to an NFA and determinize it (first exponential step), complement it and translate back to a regular expression (second exponential step). For the intersection there is a similar algorithm through a translation to NFAs, taking the crossproduct and a retranslation to a regular expression. Note that both algorithms do not only take double exponential time but also result in a regular expression of double exponential size. In this paper, we exhibit classes of regular expressions for which this double
exponential size increase cannot be avoided. Furthermore, when the number $k$ of regular expressions is fixed, $r\cap r$ can be constructed in exponential time and we prove a matching lower bound for the size increase. In addition, we consider the fragments of one-unambiguous and single-occurrence regular expressions relevant to XML schema languages [2, 3, 13, 23]. Our main results are summarized in Table 1.

The main technical part of the paper is centered around the generalization of a result by Ehrenfeucht and Zeiger [8]. They exhibit a class of languages $(Z_n)_{n \in \mathbb{N}}$ each of which can be accepted by a DFA of size $O(n^2)$ but cannot be defined by a regular expression of size smaller than $2^{n-1}$. The most direct way to define $Z_n$ is by the DFA that accepts it: the DFA is a graph consisting of $n$ states, labeled 0 to $n-1$, which are fully connected and the edge between state $i$ and $j$ carries the label $a_{i,j}$. It now accepts all paths in the graph, that is, all strings of the form $a_{i_0,i_1}a_{i_1,i_2}\cdots a_{i_k,i_{k+1}}$. Note that the alphabet over which $Z_n$ is defined grows quadratically with $n$. We generalize their result to a four-letter alphabet. In particular, we define $K_n$ as the binary encoding of $Z_n$ using a suitable encoding for $a_{i,j}$ and prove that every regular expression defining $K_n$ should be at least of size $2^n$. As integers are encoded in binary the complement and intersection of regular expressions can now be used to separately encode $K_{2^n}$ (and slight variations thereof) leading to the desired results. In [9] the same generalization as obtained here is attributed to Waizenegger [35]. Unfortunately, we believe that proof to be incorrect as we discuss in the full version of this paper.

Although the succinctness of various automata models have been investigated in depth [14] and more recently those of logics over (unary alphabet) strings [13], the succinctness of regular expressions has hardly been addressed. For the complement of a regular expression an exponential lower bound is given by Ellul et al [9]. For the intersection of an arbitrary number of regular expressions Petersen gave an exponential lower bound [28], while Ellul et al [9] mention a quadratic lower bound for the intersection of two regular expressions. In fact, in [9], it is explicitly asked what the maximum achievable blow-up is for the complement of one and the intersection of two regular expressions (Open Problems 4 and 5). Although we do not answer these questions in the most precise way, our lower bounds improve the existing ones by one exponential and are tight in the sense that the target expression can be constructed in the time class matching the space complexity of the lower bounds.

Succinctness of complement and intersection relate to the succinctness of semi-extended $(\text{RE}(\cap))$ and extended regular expressions $(\text{RE}(\cap, \neg))$. These are regular expressions augmented with intersection and both complement and intersection operators, respectively. Their membership problem has been extensively studied [18, 20, 26, 28, 30]. Furthermore, non-emptiness and equivalence of $\text{RE}(\cap, \neg)$ is non-elementary [32]. For $\text{RE}(\cap)$, inequivalence is EXPSPACE-complete [10, 14, 23], and non-emptiness is PSPACE-complete [10, 14] even when restricted to the intersection of a (non-constant) number of regular expressions [19]. Several of these papers hint upon the succinctness of the intersection operator and provide dedicated techniques in dealing with the new operator directly rather than through a translation to ordinary regular expressions [21, 22]. Our results present a double exponential lower bound in translating $\text{RE}(\cap)$ to $\text{RE}$ and therefore justify even more the development for specialized techniques.

A final motivation for this research stems from its application in the emerging area of XML-theory [21, 27, 31, 34]. From a formal language viewpoint, XML documents can be seen as labeled unranked trees and collections of these documents are defined by schemas. A schema can take various forms, but the most common ones are Document Type Definitions (DTDs) [1] and XML Schema Definitions (XSDs) [32] which are grammar based formalisms.
with regular expressions at right-hand sides of rules [23, 24]. Many questions concerning
schemas reduce to corresponding questions on the classes of regular expressions used as
right-hand sides of rules as is exemplified for the basic decision problems studied in [11]
and [22]. Furthermore, the lower bounds presented here are utilized in [14] to prove, among
other things, lower bounds on the succinctness of existential and universal pattern-based
schemas on the one hand, and single-type EDTDs (a formalization of XSDs) and DTDs,
on the other hand. As the DTD and XML Schema specification require regular expres-
sions occurring in rules to be deterministic, formalized by Brüggemann-Klein and Wood in
terms of one-unambiguous regular expressions [6], we also investigate the complement and
intersection of those. In particular, we show that a one-unambiguous regular expressions
can be complemented in polynomial time, whereas the lower bounds concerning intersection
carry over from unrestricted regular expressions. A study in [3] reveals that most of the
one-unambiguous regular expression used in practice take a very simple form: every alpha-
bet symbol occurs at most once. We refer to those as single-occurrence regular expressions
(SOREs) and show a tight exponential lower bound for intersection.

Outline. In Section 2, we introduce the necessary notions concerning (one-unambiguous)
regular expressions and automata. In Section 3, we extend the result by Ehrenfeucht and
Zeiger to a fixed alphabet using the family of languages \( (K_n)_{n \in \mathbb{N}} \). In Section 4, we consider
the succinctness of complement. In Section 5, we consider the succinctness of intersection
of several classes of regular expressions. We conclude in Section 6. A version of this paper
containing all proofs is available from the authors’ webpages.

2. Preliminaries

2.1. Regular expressions

By \( \mathbb{N} \) we denote the natural numbers without zero. For the rest of the paper, \( \Sigma \) always
denotes a finite alphabet. A \( \Sigma \)-string (or simply string) is a finite sequence \( w = a_1 \cdots a_n \)
of \( \Sigma \)-symbols. We define the length of \( w \), denoted by \( |w| \), to be \( n \). We denote the empty
string by \( \varepsilon \). The set of positions of \( w \) is \( \{1, \ldots , n\} \) and the symbol of \( w \) at position \( i \) is \( a_i \).
By \( w_1, w_2 \) we denote the concatenation of two strings \( w_1 \) and \( w_2 \). As usual, for readability,
we denote the concatenation of \( w_1 \) and \( w_2 \) by \( w_1 w_2 \). The set of all strings is denoted
by \( \Sigma^* \) and the set of all non-empty strings by \( \Sigma^+ \). A string language is a subset of \( \Sigma^* \).
For two string languages \( L, L' \subseteq \Sigma^* \), we define their concatenation \( L \cdot L' \) to be the set
\( \{w \cdot w' \mid w \in L, w' \in L'\} \). We abbreviate \( L \cdot L \cdots L \) (\( i \) times) by \( L^i \).

The set of regular expressions over \( \Sigma \), denoted by \( \text{RE} \), is defined in the usual way: \( \emptyset \),
\( \varepsilon \), and every \( \Sigma \)-symbol is a regular expression; and when \( r_1 \) and \( r_2 \) are regular expressions,
then \( r_1 \cdot r_2 \), \( r_1 + r_2 \), and \( r_1^* \) are also regular expressions.

<table>
<thead>
<tr>
<th></th>
<th>complement</th>
<th>intersection (fixed)</th>
<th>intersection (arbitrary)</th>
</tr>
</thead>
<tbody>
<tr>
<td>regular expression</td>
<td>2-exp</td>
<td>exp</td>
<td>2-exp</td>
</tr>
<tr>
<td>one-unambiguous</td>
<td>poly</td>
<td>exp</td>
<td>2-exp</td>
</tr>
<tr>
<td>single-occurrence</td>
<td>poly</td>
<td>exp</td>
<td>exp</td>
</tr>
</tbody>
</table>

Table 1: Overview of the size increase for the various operators and subclasses. All non-
polynomial complexities are tight.
By $\text{RE}(\cap, \neg)$ we denote the class of extended regular expressions, that is, $\text{RE}$ extended with intersection and complementation operators. So, when $r_1$ and $r_2$ are $\text{RE}(\cap, \neg)$-expressions then so are $r_1 \cap r_2$ and $\neg r_1$. By $\text{RE}(\cap)$ and $\text{RE}(\neg)$ we denote $\text{RE}$ extended solely with the intersection and complement operator, respectively.

The language defined by an extended regular expression $r$, denoted by $L(r)$, is inductively defined as follows: $L(\emptyset) = \emptyset$; $L(\varepsilon) = \{\varepsilon\}$; $L(a) = \{a\}$; $L(r_1 \cdot r_2) = L(r_1) \cdot L(r_2)$; $L(r_1 + r_2) = L(r_1) \cup L(r_2)$; $L(r^*) = \{\varepsilon\} \cup \bigcup_{i=1}^{\infty} L(r)^i$; $L(r_1 \cap r_2) = L(r_1) \cap L(r_2)$; and $L(\neg r_1) = \Sigma^* \setminus L(r_1)$.

By $\bigcup_{i=1}^{k} r_i$, and $r^k$, with $k \in \mathbb{N}$, we abbreviate the expression $r_1 + \cdots + r_k$, and $rr \cdots r$ ($k$-times), respectively. For a set $S = \{a_1, \ldots, a_n\} \subseteq \Sigma$, we abbreviate by $S$ the regular expression $a_1 + \cdots + a_n$.

We define the size of an extended regular expression $r$ over $\Sigma$, denoted by $|r|$, as the number of $\Sigma$-symbols and operators occurring in $r$ disregarding parentheses. This is equivalent to the length of its (parenthesis-free) reverse Polish form [37]. Formally, $|\emptyset| = |\varepsilon| = |a| = 1$, for $a \in \Sigma$, $| r_1 \cap r_2 | = |r_1 + r_2 | = |r_1| + |r_2| + 1$, and $| \neg r | = |r^*| = |r| + 1$.

Other possibilities considered in the literature for defining the size of a regular expression are: (1) counting all symbols, operators, and parentheses [1, 7]; or, (2) counting only the $\Sigma$-symbols. However, Ellul et al. [1] have shown that for regular expressions (so, without $\neg$ and $\cap$), provided they are preprocessed by syntactically eliminating superfluous $\emptyset$- and $\varepsilon$-symbols, and nested stars, the three length measures are identical up to a constant multiplicative factor. For extended regular expressions, counting only the $\Sigma$-symbols is not sufficient, since for instance the expression $(\neg \varepsilon)(\neg \varepsilon)(\neg \varepsilon)$ does not contain any $\Sigma$-symbols. Therefore, we define the size of an expression as the length of its reverse Polish form.

### 2.2. One-unambiguous regular expressions and SOREs

As mentioned in the introduction, several XML schema languages restrict regular expressions occurring in rules to be deterministic, formalized by Brüggemann-Klein and Wood [4] in terms of one-unambiguity. We introduce this notion next.

To indicate different occurrences of the same symbol in a regular expression, we mark symbols with subscripts. For instance, the marking of $(a + b)^*a + bc$ is $(a_1 + b_2)^*a_3 + b_4c_5$. We denote by $r^\circ$ the marking of $r$ and by $\text{Sym}(r^\circ)$ the subscripted symbols occurring in $r^\circ$. When $r$ is a marked expression, then $r^\circ$ over $\Sigma$ is obtained from $r$ by dropping all subscripts. This notion is extended to words and languages in the usual way.

**Definition 2.1.** A regular expression $r$ is one-unambiguous iff for all strings $w, u, v \in \text{Sym}(r^\circ)^*$, and all symbols $x, y \in \text{Sym}(r^\circ)$, the conditions $uxv, uyw \in L(r^\circ)$ and $x \neq y$ imply $x^\circ \neq y^\circ$.

For instance, the regular expression $r = a^*a$, with marking $r^\circ = a_1^*a_2$, is not one-unambiguous. Indeed, the marked strings $a_1a_2$ and $a_3a_1a_2$ both in $L(r^\circ)$ do not satisfy the conditions in the previous definition. The equivalent expression $aa^*$, however, is one-unambiguous. The intuition behind the definition is that positions in the input string can be matched in a deterministic way against a one-unambiguous regular expression without looking ahead. For instance, for the expression $aa^*$, the first $a$ of an input string is always matched against the leading $a$ in the expression, while every subsequent $a$ is matched against the last $a$. Unfortunately, one-unambiguous regular languages do not form a very robust class as they are not even closed under the Boolean operations [1].
The following subclass captures the class of regular expressions occurring in XML schemas on the Web [2]:

**Definition 2.2.** A **single-occurrence regular expression (SORE)** is a regular expression where every alphabet symbol occurs at most once. In addition, we allow the operator \( r^+ \) which defines \( rr^* \).

For instance, \((a + b)^+ c\) is a SORE while \( a^* (a + b)^+ \) is not. Clearly, every SORE is one-unambiguous. Note that SOREs define local languages and that over a fixed alphabet there are only finitely many of them.

### 2.3. Finite automata

A non-deterministic finite automaton (NFA) \( A \) is a 4-tuple \((Q, q_0, \delta, F)\) where \( Q \) is the set of states, \( q_0 \) is the initial state, \( F \) is the set of final states and \( \delta \subseteq Q \times \Sigma \times Q \) is the transition relation. We write \( q \Rightarrow_{A,w} q' \) when \( w \) takes \( A \) from state \( q \) to \( q' \). So, \( w \) is accepted by \( A \) if \( q_0 \Rightarrow_{A,w} q' \) for some \( q' \in F \). The set of strings accepted by \( A \) is denoted by \( L(A) \).

The size of an NFA is \(|Q| + |\delta|\). An NFA is deterministic (or a DFA) if for all \( a \in \Sigma, q \in Q, |\{(q, a, q') \in \delta | q' \in Q\}| \leq 1 \).

We make use of the following known results.

**Theorem 2.3.** Let \( A_1, \ldots, A_m \) be NFAs over \( \Sigma \) with \(|A_i| = n_i \) for \( i \leq m \), and \(|\Sigma| = k \).

1. A regular expression \( r \), with \( L(r) = L(A_1) \), can be constructed in time \( O(m_1 \cdot 4^{m_1}) \), where \( m_1 \) is the number of states of \( A_1 \) [24, 3].
2. A DFA \( B \) with \( 2^{n_1} \) states, such that \( L(B) = L(A_1) \), can be constructed in time \( O(2^{n_1}) \) [3].
3. A DFA \( B \) with \( 2^{m_1} \) states, such that \( L(B) = \Sigma^* \setminus L(A_1) \), can be constructed in time \( O(2^{m_1}) \) [3].
4. Let \( r \in RE \). An NFA \( B \) with \(|r| + 1 \) states, such that \( L(B) = L(r) \), can be constructed in time \( O(|r| \cdot |\Sigma|) \) [4].
5. Let \( r \in RE(\cap) \). An NFA \( B \) with \( 2^{|r|} \) states, such that \( L(B) = L(r) \), can be constructed in time exponential in the size of \( r \) [1].

### 3. A generalization of a Theorem by Ehrenfeucht and Zeiger to a fixed alphabet

We first introduce the family \((Z_n)_{n \in \mathbb{N}}\) defined by Ehrenfeucht and Zeiger over an alphabet whose size grows quadratically with the parameter \( n \) [3]:

**Definition 3.1.** Let \( n \in \mathbb{N} \) and \( \Sigma_n = \{a_{i,j} | 0 \leq i, j \leq n - 1\} \). Then, \( Z_n \) contains exactly all strings of the form \( a_{i_0,i_1}a_{i_1,i_2} \cdots a_{i_{k-1},i_k} \) where \( k \in \mathbb{N} \).

A way to interpret \( Z_n \) is to consider the DFA with states \( \{0, \ldots, n - 1\} \) which is fully connected and where the edge between state \( i \) and \( j \) is labeled with \( a_{i,j} \). The language \( Z_n \) then consists of all paths in the DFA.

Ehrenfeucht and Zeiger obtained the succinctness of DFAs with respect to regular expressions through the following theorem:

---

1Actually, in [3], only paths from state 0 to state \( n - 1 \) are considered. We use our slightly modified definition as it will be easier to generalize to a fixed arity alphabet suited for our purpose in the sequel.
Theorem 3.2. For $n \in \mathbb{N}$, any regular expression defining $Z_n$ must be of size at least $2^{n-1}$. Furthermore, there is a DFA of size $O(n^2)$ accepting $Z_n$.

Our language $K_n$ is then the straightforward binary encoding of $Z_n$ that additionally swaps the pair of indices in every symbol $a_{i,j}$. Thereto, for $a_{i,j} \in \Sigma_n$, define the function $\rho_n$ as

$$\rho_n(a_{i,j}) = \text{enc}(j)\text{enc}(i)\#,$$

where $\text{enc}(i)$ and $\text{enc}(j)$ denote the $\lceil \log(n) \rceil$-bit binary encodings of $i$ and $j$, respectively. Note that since $i, j < n$, $i$ and $j$ can be encoded using only $\lceil \log(n) \rceil$-bits. We extend the definition of $\rho_n$ to strings in the usual way: $\rho_n(a_{i_0,i_1} \cdots a_{i_{k-1},i_k}) = \rho_n(a_{i_0,i_1}) \cdots \rho_n(a_{i_{k-1},i_k})$.

We are now ready to define $K_n$.

Definition 3.3. Let $\Sigma_K = \{0, 1, $, $\#\}$. For $n \in \mathbb{N}$, let $K_n = \{\rho_n(w) \mid w \in Z_n\}$.

For instance, for $n = 5$, $w = a_{3,2}a_{2,1}a_{1,4}a_{4,2} \in Z_5$ and thus $\rho_n(w) = 010\#011\#001\#010\#100\#100\#$ $\in K_5$.

We generalize the previous theorem as follows:

Theorem 3.4. For any $n \in \mathbb{N}$, with $n \geq 2$,

(1) any regular expression defining $K_n$ is of size at least $2^n$; and,

(2) there is a DFA $A_n$ of size $O(n^2 \log n)$ defining $K_n$.

The construction of $A_n$ is omitted. The rest of this section is devoted to the proof of Theorem 3.4(1). It follows the structure of the proof of Ehrenfeucht and Zeiger but is technically more involved as it deals with binary encodings of integers.

We start by introducing some terminology. Let $w = a_{i_0,i_1}a_{i_1,i_2} \cdots a_{i_{k-1},i_k} \in Z_n$. We say that $i_0$ is the start-point of $w$ and $i_k$ is its end-point. Furthermore, we say that $w$ contains $i$ or $j$ occurs in $w$ if $i$ occurs as an index of some symbol in $w$. That is, $a_{i,j}$ or $a_{j,i}$ occurs in $w$ for some $j$. For instance, $a_{0,2}a_{2,2}a_{2,1} \in Z_5$, has start-point 0, end-point 1, and contains 0, 1 and 2. The notions of contains, occurs, start- and end-point of a string $w$ are also extended to $K_n$. So, the start and end-points of $\rho_n(w)$ are the start and end-points of $w$, and $w$ contains the same integers as $\rho_n(w)$.

For a regular expression $r$, we say that $i$ is a sidekick of $r$ when it occurs in every non-empty string defined by $r$. A regular expression $s$ is a starred subexpression of a regular expression $r$ when $s$ is a subexpression of $r$ and is of the form $t^*$. Now, the following lemma holds:

Lemma 3.5. Any starred subexpression $s$ of a regular expression $r$ defining $K_n$ has a sidekick.

We now say that a regular expression $r$ is normal if every starred subexpression of $r$ has a sidekick. In particular, any expression defining $K_n$ is normal. We say that a regular expression $r$ covers a string $w$ if there exist strings $u, u' \in \Sigma^*$ such that $uwu' \in L(r)$. If there is a greatest integer $m$ for which $r$ covers $w^m$, we call $m$ the index of $w$ in $r$ and denote it by $I_n(r)$. In this case we say that $r$ is $w$-finite. Otherwise, we say that $r$ is $w$-infinite. The index of a regular expression can be used to give a lowerbound on its size according to the following lemma.
Lemma 3.6 ([8]). For any regular expression \( r \) and string \( w \), if \( r \) is \( w \)-finite, then \( I_w(r) < 2|r| \).

Now, we can state the most important property of \( K_n \).

Lemma 3.7. Let \( n \geq 2 \). For any \( C \subseteq \{0, \ldots, n - 1\} \) of cardinality \( k \) and \( i \in C \), there exists a string \( w \in K_n \) with start- and end-point \( i \) only containing integers in \( C \), such that any normal regular expression \( r \) which covers \( w \) is of size at least \( 2^k \).

Proof. The proof is by induction on the value of \( k \). For \( k = 1 \), \( C = \{i\} \). Then, define \( w = \text{enc}(i)$\text{enc}(i)$\text{#} \), which satisfies all conditions and any expression covering \( w \) must definitely have a size of at least 2.

For the inductive step, let \( C = \{j_1, \ldots, j_k\} \). Define \( C_\ell = C \setminus \{j_{\ell \mod k + 1}\} \) and let \( w_\ell \) be the string given by the induction hypothesis with respect to \( C_\ell \) (of size \( k - 1 \)) and \( j_\ell \). Note that \( j_\ell \in C_\ell \). Further, define \( m = 2^{k+1} \) and set
\[
\begin{align*}
  w &= \text{enc}(j_1)\text{enc}(i)\#w_m^1\text{enc}(j_2)\text{#enc}(j_1)\#w_m^2\text{enc}(j_2)\text{#} \cdots \text{#}w_m^n\text{enc}(i)\text{#enc}(j_k)\#.
\end{align*}
\]

Then, \( w \in K_n \), has \( i \) as start and end-point and only contains integers in \( C \). It only remains to show that any expression \( r \) which is normal and covers \( w \) is of size at least \( 2^k \).

Fix such a regular expression \( r \). If \( r \) is \( w_\ell \)-finite for some \( \ell \leq k \), then, \( I_{w_\ell}(r_k) \geq m = 2^{k+1} \) by construction of \( w \). By Lemma 3.3, \( |r| \geq 2^k \) and we are done.

Therefore, assume that \( r \) is \( w_\ell \)-finite for every \( \ell \leq k \). For every \( \ell \leq k \), consider all subexpressions of \( r \) which are \( w_\ell \)-finite. It is easy to see that all minimal elements in this set of subexpressions must be starred subexpressions. Here and in the following, we say that an expression is minimal with respect to a set simply when no other expression in the set is a subexpression. Indeed, a subexpression of the form \( a \) or \( \varepsilon \) can never be \( w_\ell \)-finite and a subexpression of the form \( r_1r_2 \) or \( r_1 + r_2 \) can only be \( w_\ell \)-finite if \( r_1 \) and/or \( r_2 \) are \( w_\ell \)-finite and is thus not minimal with respect to \( w_\ell \)-infinity. Among these minimal starred subexpressions for \( w_\ell \), choose one and denote it by \( s_\ell \). Let \( E = \{s_1, \ldots, s_k\} \). Note that since \( r \) is normal, all its subexpressions are also normal. As in addition each \( s_\ell \) covers \( w_\ell \), by the induction hypothesis the size of each \( s_\ell \) is at least \( 2^{k-1} \). Now, choose from \( E \) some expression \( s_\ell \) such that \( s_\ell \) is minimal with respect to the other elements in \( E \).

As \( r \) is normal and \( s_\ell \) is a starred subexpression of \( r \), there is an integer \( j \) such that every non-empty string in \( L(s_\ell) \) contains \( j \). By definition of the strings \( w_1, \ldots, w_k \), there is some \( w_p, p \leq k \), such that \( w_p \) does not contain \( j \). Denote by \( s_p \) the starred subexpression from \( E \) which is \( w_p \)-infinite. In particular, \( s_\ell \) and \( s_p \) cannot be the same subexpression of \( r \).

Now, there are three possibilities:

- \( s_\ell \) and \( s_p \) are completely disjoint subexpressions of \( r \). That is, they are both not a subexpression of one another. By induction they must both be of size \( 2^{k-1} \) and thus \( |r| \geq 2^{k-1} + 2^{k-1} = 2^k \).
- \( s_p \) is a strict subexpression of \( s_\ell \). This is not possible since \( s_\ell \) is chosen to be a minimum element from \( E \).
- \( s_\ell \) is a strict subexpression of \( s_p \). We show that if we replace \( s_\ell \) by \( \varepsilon \) in \( s_p \), then \( s_p \) is still \( w_p \)-infinite. It then follows that \( s_p \) still covers \( w_\ell \) and thus \( s_p \) without \( s_\ell \) is of size at least \( 2^k - 1 \). As \( |s_\ell| \geq 2^k - 1 \) as well it follows that \( |r| \geq 2^k \).

\[\text{In fact, in [8] the length of an expression is defined as the number of } \Sigma \text{-symbols occurring in it. However, since our length measure also contains these } \Sigma \text{-symbols, this lemma still holds in our setting.}\]
To see that $s_p$ without $s_\ell$ is still $w_p$-infinite, recall that any non-empty string defined by $s_\ell$ contains $j$ and $j$ does not occur in $w_p$. Therefore, a full iteration of $s_\ell$ can never contribute to the matching of any number of repetitions of $w_p$. So, $s_p$ can only lose its $w_p$-infinity by this replacement if $s_\ell$ contains a subexpression which is itself $w_p$-infinite. However, this then also is a subexpression of $s_p$ and $s_p$ is chosen to be minimal with respect to $w_p$-infinity, a contradiction. We can only conclude that $s_p$ without $s_\ell$ is still $w_p$-infinite.

Since by Lemma 3.3 any expression defining $K_n$ is normal, Theorem 3.4(1) directly follows from Lemma 3.7 by choosing $i = 0$, $k = n$. This concludes the proof of Theorem 3.4(1).

4. Complementing regular expressions

It is known that extended regular expressions are non-elementary more succinct than classical ones [7, 33]. Intuitively, each exponent in the tower requires nesting of an additional complement. In this section, we show that in defining the complement of a single regular expression, a double-exponential size increase cannot be avoided in general. In contrast, when the expression is one-unambiguous its complement can be computed in polynomial time.

**Theorem 4.1.** (1) For every regular expression $r$ over $\Sigma$, a regular expression $s$ with $L(s) = \Sigma^* \setminus L(r)$ can be constructed in time $O(2^{|r|+1} \cdot |\Sigma| \cdot 4^{|r|+1})$.

(2) Let $\Sigma$ be a four-letter alphabet. For every $n \in \mathbb{N}$, there is a regular expressions $r_n$ of size $O(n)$ such that any regular expression $r$ defining $\Sigma^* \setminus L(r_n)$ is of size at least $2^{2^n}$.

**Proof.** (2) Take $\Sigma$ as $\Sigma_K$, that is, $\{0, 1, $, #\}. Let $n \in \mathbb{N}$. We define an expression $r_n$ of size $O(n)$, such that $\Sigma^* \setminus L(r_n) = K_{2^n}$. By Theorem 3.4, any regular expression defining $K_{2^n}$ is of size exponential in $2^n$, that is, of size $2^{2^n}$. By $r^{[0,n-1]}$ we abbreviate the expression $(\varepsilon + r(\varepsilon + r(\varepsilon \cdots (\varepsilon + r)))$, with a nesting depth of $n - 1$. We then define $r_n$ as the disjunction of the following expressions:

- all strings that do not start with a prefix in $(0 + 1)^n$:
  $$\Sigma^{[0,n]} + (0 + 1)^{[0,n-1]}(\# + \#)\Sigma^* + (0 + 1)^n(0 + 1 + \#)\Sigma^*$$
- all strings where a $\#$ is not followed by a string in $(0 + 1)^n$:
  $$\Sigma^*\Sigma^{[0,n-1]}(\# + \#) + (0 + 1)^n\#\Sigma^*$$
- all strings where a non-final $\#$ is not followed by a string in $(0 + 1)^n$:
  $$\Sigma^*\Sigma^{[0,n-1]}(\# + \#) + (0 + 1)^n\#\Sigma^*$$
- all strings that do not end in $\#$:
  $$\Sigma^*(0 + 1 + \#)$$
- all strings where the corresponding bits of corresponding blocks are different:
  $$((0 + 1)^* + \Sigma^*\#(0 + 1)^*)0\Sigma^{[0,n-1]}2\Sigma^* + ((0 + 1)^* + \Sigma^*\#(0 + 1)^*)1\Sigma^{[0,n-1]}2\Sigma^*.$$ 

It should be clear that a string over $\{0, 1, $, #\}$ is matched by none of the above expressions if and only if it belongs to $K_{2^n}$. So, the complement of $r_n$ defines exactly $K_{2^n}$. 
\[\Box\]
The previous theorem essentially shows that in complementing a regular expression, there is no better algorithm than translating to a DFA, computing the complement and translating back to a regular expression which includes two exponential steps. However, when the given regular expression is one-unambiguous, a corresponding DFA can be computed in quadratic time through the Glushkov construction eliminating already one exponential step. The proof of the next theorem shows that the complement of that DFA can be directly defined by a regular expression of polynomial size.

**Theorem 4.2.** For any one-unambiguous regular expression $r$ over an alphabet $\Sigma$, a regular expression $s$ defining $\Sigma^* \setminus L(r)$ can be constructed in time $O(n^3)$, where $n$ is the size of $r$.

**Proof.** Let $r$ be a one-unambiguous expression over $\Sigma$. We introduce some notation.

- **init**($r$) contains all $\Sigma$-symbols which are not the first symbol in any word defined by $r$, that is, Not-First($r$) = $\Sigma \setminus \{a \mid a \in \Sigma \land \exists w \in \Sigma^*, aw \in L(r)\}$.
- For any symbol $x \in \text{Sym}(r^\flat)$, the set Not-Follow($r, x$) contains all $\Sigma$-symbols of which no marked version can follow $x$ in any word defined by $r^\flat$. That is, Not-Follow($r, x$) = $\Sigma \setminus \{y \sharp \mid y \in \text{Sym}(r^\flat) \land \exists w, w' \in \text{Sym}(r^\flat)^*, wxyw' \in L(r^\flat)\}$.
- The set Last($r$) contains all marked symbols which are the last symbol of some word defined by $r^\flat$. Formally, Last($r$) = $\{x \mid x \in \text{Sym}(r^\flat) \land \exists w \in \Sigma^*, wx \in L(r^\flat)\}$.

We define the following regular expressions:

- $\text{init}(r) = \begin{cases} \text{Not-First}(r)\Sigma^* & \text{if } \varepsilon \in L(r); \\
\varepsilon + \text{Not-First}(r)\Sigma^* & \text{if } \varepsilon \notin L(r). \end{cases}$
- For every $x \in \text{Sym}(r^\flat)$, let $r^\sharp_x$ be the expression defining $\{wx \mid w \in \text{Sym}(r^\flat)^* \land \exists u \in \text{Sym}(r^\flat)^*, wxu \in L(r^\flat)\}$. That is, all prefixes of strings in $r^\flat$ ending in $x$. Then, let $r_x$ define $L(r_x^\sharpg)$.

We are now ready to define $s$:  

$$\text{init}(r) + \bigcup_{x \notin \text{Last}(r)} r_x(\varepsilon) + \text{Not-Follow}(r,x)\Sigma^* + \bigcup_{x \in \text{Last}(r)} r_x\text{Not-Follow}(r,x)\Sigma^*.$$  

It can be shown that $s$ can be constructed in time cubic in the size of $r$ and that $s$ defines the complement of $r$. The latter is proved by exhibiting a direct correspondence between $s$ and the complement of the Glushkov automaton of $r$.

We conclude this section by remarking that one-unambiguous regular expressions are not closed under complement and that the constructed $s$ is therefore not necessarily one-unambiguous.

### 5. Intersecting regular expressions

In this section, we study the succinctness of intersection. In particular, we show that the intersection of two (or any fixed number) and an arbitrary number of regular expressions are exponentially and double exponentially more succinct than regular expressions, respectively. Actually, the exponential bound for a fixed number of expressions already holds for single-occurrence regular expressions, whereas the double exponential bound for an arbitrary number of expressions only carries over to one-unambiguous expressions. For single-occurrence expressions this can again be done in exponential time.

In this respect, we introduce a slightly altered version of $K_n$.  

---
Definition 5.1. Let $\Sigma_L = \{0, 1, \$ , \#, \triangle\}$. For all $n \in \mathbb{N}$, $L_n = \{\rho_n(w)\triangle \mid w \in Z_n \land |w| \text{ is even}\}$.

We also define a variant of $Z_n$ which only slightly alters the $a_{i,j}$ symbols in $Z_n$. Thereto, let $\Sigma^n_\circ = \{a^\circ_{i,j}, a^\circ_{i,j} \mid 0 \leq i, j < n\}$ and set $\hat{\rho}(a^\circ_{i,j}a^\circ_{j,k}) = \triangle a^\circ_{i,j}a^\circ_{j,k}$ and $\hat{\rho}(a^\circ_{i_0,i}a^\circ_{i_1,i_2} \cdots a^\circ_{i_{k-2},i_{k-1}}a^\circ_{i_{k-1},i_k}) = \hat{\rho}(a_{i_0,i}a^\circ_{i_1,i_2} \cdots a^\circ_{i_{k-2},i_{k-1}}a_{i_{k-1},i_k})$, where $k$ is even.

Definition 5.2. Let $n \in \mathbb{N}$ and $\Sigma^n_M = \Sigma^n_\circ \cup \{\triangle_0, \triangle_0, \ldots, \triangle_{n-1}, \triangle_{n-2}\}$. Then, $M_n = \{\hat{\rho}(w)\triangle_i \mid w \in Z_n \land |w| \text{ is even} \land i \text{ is the end-point of } w\}$.

Note that paths in $M_n$ are those in $Z_n$ where every odd position is promoted to a circled one ($^\circ$), and triangles labeled with the non-circled positions are added. For instance, the string $a_2^\circ a_4^\circ a_3^\circ a_3^\circ a_3^\circ a_3^\circ \epsilon_0 \in Z_5$ is mapped to the string $\triangle_2^2 a_2^\circ a_4^\circ \epsilon_3 \triangle_3^3 a_3^\circ a_3^\circ \epsilon_3 a_3^\circ \epsilon_0 \triangle_0 \in M_5$.

We make use of the following property:

Lemma 5.3. Let $n \in \mathbb{N}$.

1. Any regular expression defining $L_n$ is of size at least $2^n$.
2. Any regular expression defining $M_n$ is of size at least $2^{n-1}$.

The next theorem shows the succinctness of the intersection operator.

Theorem 5.4. (1) For any $k \in \mathbb{N}$ and regular expressions $r_1, \ldots, r_k$, a regular expression defining $\bigcap_{i=1}^{k} L(r_i)$ can be constructed in time $O((m+1)^k \cdot |\Sigma| \cdot 4^{(m+1)^k})$, where $m = \max \{|r_i| \mid 1 \leq i \leq k\}$.

(2) For every $n \in \mathbb{N}$, there are SOREs $r_n$ and $s_n$ of size $O(n^2)$ such that any regular expression defining $L(r_n) \cap L(s_n)$ is of size at least $2^{n-1}$.

(3) For each $r \in RE(\cap)$ an equivalent regular expression can be constructed in time $O(2^{2^{|r|}} \cdot |\Sigma| \cdot 4^{2^{2^{|r|}}})$.

(4) For every $n \in \mathbb{N}$, there are one-unambiguous regular expressions $r_1, \ldots, r_m$, with $m = 2n+1$, of size $O(n)$ such that any regular expression defining $\bigcap_{i=1}^{m} L(r_i)$ is of size at least $2^n$.

(5) Let $r_1, \ldots, r_n$ be SOREs. A regular expression defining $\bigcap_{i=1}^{n} L(r_i)$ can be constructed in time $O(m \cdot |\Sigma| \cdot 4^m)$, where $m = \sum_{i=1}^{n} |r_i|$.

Proof. (2) Let $n \in \mathbb{N}$. By Lemma 5.3, any regular expression defining $M_n$ is of size at least $2^{n-1}$. We define SOREs $r_n$ and $s_n$ of size quadratic in $n$, such that $L(r_n) \cap L(s_n) = M_n$.

We start by partitioning $\Sigma^n_M$ in two different ways. To this end, for every $i < n$, define $Out_i = \{a^\circ_{i,j} \mid 0 \leq j < n\}$, $In_i = \{a_{j,i} \mid 0 \leq j < n\}$, $Out_{\epsilon} = \{a^\circ_{i,j} \mid 0 \leq j < n\}$, and, $In_{\epsilon} = \{a_{j,i} \mid 0 \leq j < n\}$. Then,

$$\Sigma^n_M = \bigcup_{i} In_i \cup Out_i \cup \{\triangleright_i, \triangle_i\} = \bigcup_{i} In_{\epsilon} \cup Out_{\epsilon} \cup \{\triangleright_i, \triangle_i\}.$$ 

Further, define

$$r_n = (\triangleright_0 + \cdots + \triangleright_{n-1}) \bigcup_{\epsilon} In_{\epsilon} Out_{\epsilon}^+ (\triangle_0 + \cdots + \triangle_{n-1})$$

and

$$s_n = \left(\bigcup_{i} (In_i + \epsilon) (\triangleright_i + \triangle_i) (Out_i + \epsilon)\right)^\cdot.$$ 

Now, $r_n$ checks that every string consists of a sequence of blocks of the form $\triangleright_i a_{j,k}$, $a_{k,e,\ell}$, for $i, j, k, \ell < n$, ending with a $\triangle_i$, for $i < n$. It thus sets the format of the strings
checks whether the circled indices are equal. Further, $s_n$ checks whether the non-circled indices are equal and whether the triangles have the correct indices. Since the alphabet of $M_n$ is of size $O(n^2)$, also $r_n$ and $s_n$ are of size $O(n^2)$.

(4) Let $n \in \mathbb{N}$. We define $m = 2n + 1$ one-unambiguous regular expressions of size $O(n)$, such that their intersection defines $L_{2n}$. By Lemma 5.3(1), any regular expression defining $L_{2n}$ is of size at least $2^{2n}$ and the theorem follows. For ease of readability, we denote $\Sigma_L$ simply by $\Sigma$. The expressions are as follows. There should be an even length sequence of blocks:

$$(0 + 1)^n \#(0 + 1)^n \#(0 + 1)^n \#(0 + 1)^n \#)^* \triangle.$$ 

For all $i \in \{0, \ldots, n-1\}$, the $(i+1)$th bit of the two numbers surrounding an odd $\#$ should be equal:

$$(\Sigma^i(0\Sigma^{3n+2}0 + 1\Sigma^{3n+2})\Sigma^{n-i-1} \#)^* \triangle.$$ 

For all $i \in \{0, \ldots, n-1\}$, the $(i+1)$th bit of the two numbers surrounding an even $\#$ should be equal:

$$\Sigma^{2n+2}\left(\Sigma^i(0\Sigma^{2n-i+1}(\triangle + \Sigma^{n+i+1}0\Sigma^{n-i-1} \#) + (1\Sigma^{2n-i+1}(\triangle + \Sigma^{n+i+1}\Sigma^{n-i-1} \#)))\right)^*.$$ 

Clearly, the intersection of the above expressions defines $L_{2n}$. Furthermore, every expression is of size $O(n)$ and is one-unambiguous as the Glushkov construction translates them into a DFA.

6. Conclusion

In this paper we showed that the complement and intersection of regular expressions are double exponentially more succinct than ordinary regular expressions. For complement, complexity can be reduced to polynomial for the class of one-unambiguous regular expressions although the obtained expressions could fall outside that class. For intersection, restriction to SOREs reduces complexity to exponential. It remains open whether there are natural classes of regular expressions for which both the complement and intersection can be computed in polynomial time.

Acknowledgment. We thank Juraj Hromkovič for sending us reference [3].

References


This work is licensed under the Creative Commons Attribution-NoDerivs License. To view a copy of this license, visit [http://creativecommons.org/licenses/by-nd/3.0/](http://creativecommons.org/licenses/by-nd/3.0/).