Succinctness of Regular Expressions with Interleaving, Intersection and Counting

Succinctness of Regular Expressions with Interleaving, Intersection and Counting

Wouter Gelade*

Hasselt University and Transnational University of Limburg
School for Information Technology
wouter.gelade@uhasselt.be

Abstract. Studying the impact of operations, such as intersection and interleaving, on the succinctness of regular expressions has recently received renewed attention [12–14]. In this paper, we study the succinctness of regular expressions (REs) extended with interleaving, intersection and counting operators. We show that in a translation from REs with interleaving to standard regular expressions a double exponential size increase can not be avoided. We also consider the complexity of translations to finite automata. We give a tight exponential lower bound on the translation of REs with intersection to NFAs, and, for each of the three classes of REs, we show that in a translation to a DFA a double exponential size increase can not be avoided. Together with known results, this gives a complete picture of the complexity of translating REs extended with interleaving, intersection or counting into (standard) regular expressions, NFAs, and DFAs.

1 Introduction

Regular expressions are used in many applications such as text processors, programming languages [30], and XML schema languages [5, 28]. These applications, however, usually do not restrict themselves to the standard regular expression using disjunction ($+$), concatenation ($\cdot$) and star (*), but also allow the use of additional operators. Although these operators mostly do not increase the expressive power of the regular expressions, they can have a drastic impact on succinctness, thus making them harder to handle. For instance, it is well known that expressions extended with the complement operator can describe certain languages non-elementary more succinct than standard regular expressions or finite automata [29].

In this paper, we study the succinctness of regular expressions extended with counting (RE(#)), intersection (RE($\cap$)), and interleaving (RE(&)) operators. The counting operator allows for expressions such as $a^{[2,5]}$, specifying that there must occur at least two and at most 5 a's. These RE(#)s are used in egrep [16] and Perl [30] patterns and in the XML schema language XML Schema [28]. The class RE($\cap$) is a well studied extension of the regular expressions, and is often

* Research Assistant of the Fund for Scientific Research - Flanders (Belgium)

referred to as the semi-extended regular expressions. The interleaving operator allows for expressions such as $a \,\&\, b \,\&\, c$, specifying that a, b, and c may occur in any order, and is used, for instance, in the XML schema language Relax NG [5].

A problem we consider, is the translation of extended regular expressions into (standard) regular expressions. For RE(#) and RE($\cap$) the complexity of this translation has already been settled and is exponential [18] and double exponential [12], respectively. We show that also in constructing an expression for the interleaving of a set of expressions (an hence also for an RE(&)) a double exponential size increase can not be avoided. This is the main technical result of the paper. Apart from a pure mathematical interest, the latter result has two important consequences. First, it prohibits an efficient translation from Relax NG (which allows interleaving) to XML Schema Definitions (which does not). However, as XML Schema is the widespread W3C standard, and Relax NG is a more flexible alternative, such a translation would be more than desirable. A second consequence concerns the automatic discovery of regular expression describing a set of given strings. The latter problem occurs in the learning of XML schema languages [1–3]. At present these algorithms do not take into account the interleaving operator, but for Relax NG this would be wise as this would allow to learn significantly smaller expressions.

We recently learned that Gruber and Holzer independently obtained a similar result [Personal communication]. They show that any regular expression defining the language $(a_1b_1)^* \& \cdots \& (a_nb_n)^*$ must be of size at least double exponential in n. Compared to the result in this paper, this gives a tighter bound ($2^{2^{\Omega(n)}}$ instead of $2^{2^{\Omega(\sqrt{n})}}$), and shows that the double exponential size increase already occurs for very simple expressions. On the other, the alphabet of the counterexamples grows linear with n, whereas the alphabet size is constant for the languages in this paper. Hence, the two results nicely complement each other.

We also consider the translation of extended regular expressions to NFAs. For the standard regular expressions, it is well known that such a translation can be done efficiently [4]. Therefore, when considering problems such as membership, equivalence, and inclusion testing for regular expressions the first step is almost invariantly a translation to a finite automaton. For extended regular expressions, such an approach is less fruitful. We show that an RE(&, $\cap$, #) can be translated in exponential time into an NFA. However, it has already been shown by Kilpelainen and Tuhkanen [18] and Mayer and Stockmeyer [20] that such an exponential size increase can not be avoided for RE(#) and RE(&), respectively. For the translation from RE($\cap$) to NFAs, a $2^{\Omega(\sqrt{n})}$ lower bound is reported in [25], which we here improve to $2^{\Omega(n)}$.

As the translation of extended regular expressions to NFAs already involves an exponential size increase, it is natural to ask what the size increase for DFAs is. Of course, we can translate any NFA into a DFA in exponential time, thus giving a double exponential translation, but can we do better? For instance, from the results in [12] we can conclude that given a set of regular expressions, constructing an NFA for their intersection can not avoid an exponential size increase. However, it is not too hard to see that also a DFA of exponential size

accepting their intersection can be constructed. In the present paper, we show that this is not possible for the classes RE(#), RE(∩), and RE(&). For each class we show that in a translation to a DFA, a double exponential size increase can not be avoided. An overview of all results is given in Figure 1(a).

	NFA	DFA	RE
RE(#)	$2^{\Omega(n)}$ [18]	$2^{2^{\Omega(n)}}$ (Th. 3)	$2^{\theta(n)}$ [18]
RE(∩)	$2^{\Omega(n)}$ (Pr. 2)	$2^{2^{\Omega(n)}}$ (Th. 4)	$2^{2^{\Omega(\sqrt{n})}}$ [12]
RE(&)	$2^{\Omega(n)}$ [20]	$2^{2^{\Omega(\sqrt{n})}}$ (Th. 5)	$2^{2^{\Omega(\sqrt{n})}}$ (Th. 6)
RE(&, ∩, #)	$2^{\mathcal{O}(n)}$ (Pr. 1)	$2^{2^{\mathcal{O}(n)}}$ (Pr. 3)	$2^{2^{\mathcal{O}(n)}}$ (Pr. 4)

(a)

	RE
RE ∩ RE	$2^{\Omega(n)}$ [13]
$\bigcap$ RE	$2^{2^{\Omega(\sqrt{n})}}$ [12]
RE & RE	$2^{\Omega(n)}$ [13]

(b)

Fig. 1. Table (a) gives the complexity of translating extended regular expressions into NFAs, DFAs, and regular expressions. Proposition and theorem numbers are given in brackets. Table (b) lists some related results obtained in [12] and [13]

Related work. The different classes of regular expressions considered here have been well studied. In particular, the RE(∩) and its membership [17, 19, 25] and equivalence and emptiness [10, 24, 26] problems, but also the classes RE(#) [18, 23] and RE(&) [11, 20] have received interest. Succinctness of regular expressions has been studied by Ehrenfeucht and Zeiger [8] and, more recently, by Ellul et. al [9], Gelade and Neven [12], Gruber and Holzer [13, 14], and Gruber and Johannsen [15]. Some relevant results of these papers are listed in Figure 1(b). Schott and Spehner give lower bounds for the translation of the interleaving of words to DFAs [27]. Also related, but different in nature, are the results on state complexity [32], in which the impact of the application of different operations on finite automata is studied.

Outline. In Section 2 we give the necessary definitions and present some basic results. In Sections 3, 4, and 5 we study the translation of extended regular expressions to NFAs, DFAs, and regular expressions, respectively. A version of this paper containing all proofs is available from the webpage of the author.

2 Definitions and Basic Results

2.1 Regular Expressions

By $\mathbb{N}$ we denote the natural numbers without zero. For the rest of the paper, Σ always denotes a finite alphabet. A *Σ-string* (or simply string) is a finite sequence $w = a_1 \cdots a_n$ of Σ-symbols. We define the length of w, denoted by $|w|$, to be n. We denote the empty string by ε. The set of *positions of w* is $\{1, \ldots, n\}$ and the *symbol of w at position i* is a_i. By $w_1 \cdot w_2$ 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 Σ^*. A *string language* is a

subset of Σ^*. For two string languages $L, L' \subseteq \Sigma^*$, we define their concatenation $L \cdot L'$ to be the set $\{ww' \mid w \in L, w' \in L'\}$. We abbreviate $L \cdot L \cdots L$ (i times) by L^i. By $w_1 \,\&\, w_2$ we denote the set of strings that is obtained by *interleaving* w_1 and w_2 in every possible way. That is, for $w \in \Sigma^*$, $w \,\&\, \varepsilon = \varepsilon \,\&\, w = \{w\}$, and $aw_1 \,\&\, bw_2 = (\{a\}(w_1 \,\&\, bw_2)) \cup (\{b\}(aw_1 \,\&\, w_2))$. The operator $\&$ is then extended to languages in the canonical way.

The set of *regular expressions* over Σ, denoted by RE, is defined in the usual way: $\emptyset$, ε, and every Σ-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. By $\mathrm{RE}(\&, \cap, \#)$ we denote the class of *extended regular expressions*, that is, REs extended with interleaving, intersection and counting operators. So, when r_1 and r_2 are $\mathrm{RE}(\&, \cap, \#)$-expressions then so are $r_1 \,\&\, r_2$, $r_1 \cap r_2$, and $r_1^{[k,\ell]}$ for $k, \ell \in \mathbb{N}$ with $k \leq \ell$. By $\mathrm{RE}(\&)$, $\mathrm{RE}(\cap)$, and $\mathrm{RE}(\#)$, we denote RE extended solely with the interleaving, intersection and counting operator, respectively.

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

By r^+, $\bigcup_{i=1}^{k} r_i$, and r^k, with $k \in \mathbb{N}$, we abbreviate the expression rr^*, $r_1 + \cdots + r_k$, 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 $a_1 + \cdots + a_n$. When $r^{[k,l]}$ is used in a standard regular expression, this is an abbreviation for $r^k(r + \varepsilon)^{l-k}$.

We define the *size* of an extended regular expression r over Σ, denoted by $|r|$, as the number of Σ-symbols and operators occurring in r plus the sizes of the binary representations of the integers. Formally, $|\emptyset| = |\varepsilon| = |a| = 1$, for $a \in \Sigma$, $|r_1 r_2| = |r_1 \cap r_2| = |r_1 + r_2| = |r_1 \,\&\, r_2| = |r_1| + |r_2| + 1$, $|r^*| = |r| + 1$, and $|r^{[k,\ell]}| = |r| + \lceil \log k \rceil + \lceil \log \ell \rceil$.

Intuitively, the *star height* of a regular expression r, denoted by $\mathrm{sh}(r)$ equals the number of nested stars in r. Formally, $\mathrm{sh}(\emptyset) = \mathrm{sh}(\varepsilon) = \mathrm{sh}(a) = 0$, for $a \in \Sigma$, $\mathrm{sh}(r_1 r_2) = sh(r_1 + r_2) = \max\{\mathrm{sh}(r_1), \mathrm{sh}(r_2)\}$, and $\mathrm{sh}(r^*) = \mathrm{sh}(r) + 1$. The star height of a regular language L, denoted by $\mathrm{sh}(L)$, is the minimal star height among all regular expressions defining L.

The latter two concepts are related through the following theorem due to Gruber and Holzer [13], which will allow us to reduce our questions about the size of regular expressions to questions about the star height of regular languages.

Theorem 1 ([13]). *Let L be a regular language. Then any regular expression defining L is of size at least $2^{\frac{1}{3}(sh(L)-1)} - 1$.*

2.2 Finite Automata and Graphs

A non-deterministic finite automaton (NFA) A is a 4-tuple (Q, q_0, δ, 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. As usual, we denote by $\delta^* \subseteq Q \times \Sigma^* \times Q$ the reflexive-transitive closure of δ. Then, w is accepted by A if $(q_0, w, q_f) \in \delta^*$

for some $q_f \in F$. 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 $a \in \Sigma, q \in Q$, $|\{(q, a, q') \in \delta \mid q' \in Q\}| \leq 1$.

A state $q \in Q$ is *useful* if there exist strings $w, w' \in \Sigma^*$ such that $(q_0, w, q) \in \delta^*$, and $(q, w', q_f) \in \delta^*$, for some $q_f \in F$. An NFA is *trim* if it only contains useful states. For $q \in Q$, let $\text{symbols}(q) = \{a \mid \exists p \in Q, (p, a, q) \in \delta\}$. Then, A is *state-labeled* if for any $q \in Q$, $|\text{symbols}(q)| \leq 1$, i.e., all transitions to a single state are labeled with the same symbol. In this case, we also denote this symbol by $\text{symbol}(q)$. Further, A is *non-returning* if $\text{symbols}(q_0) = \emptyset$, i.e., q_0 has no incoming transitions. A language L is *bideterministic* if there exists a DFA A, accepting L, such that the inverse of A is again deterministic. That is, A may have at most one final state and the automaton obtained by inverting every transition in A, and exchanging the initial and final state, is again deterministic.

A (directed) *graph* G is a tuple (V, E), where V is the set of *vertices* and $E \subseteq V \times V$ is the set of *edges*. A graph (U, F) is a *subgraph* of G if $U \subseteq V$ and $F \subseteq E$. For a set of vertices $U \subseteq V$, the *subgraph of G induced by U*, denoted by $G[U]$, is the graph (U, F), where $F = \{(u, v) \mid u, v \in U \wedge (u, v) \in E\}$.

A graph $G = (V, E)$ is *strongly connected* if for every pair of vertices $u, v \in V$, both u is reachable from v, and v is reachable from u. A set of edges $V' \subseteq V$ is a *strongly connected component (SCC)* of G if $G[V']$ is strongly connected and for every set V'', with $V' \subsetneq V''$, $G[V'']$ is not strongly connected.

We now introduce the *cycle rank* of a graph $G = (V, E)$, denoted by $\text{cr}(G)$, which is a measure for the structural complexity of G. It is inductively defined as follows: (1) if G is acyclic or empty, then $\text{cr}(G) = 0$, otherwise (2) if G is strongly connected, then $\text{cr}(G) = \min_{v \in V} \text{cr}(G[V \setminus \{v\}]) + 1$, and otherwise (3) $\text{cr}(G) = \max_{V'} \text{SCC of } G \, \text{cr}(G[V'])$.

Let $A = (Q, q_0, \delta, F)$ be an NFA. The *underlying graph* G of A is the graph obtained by removing the labels from the transition edges of A, or more formally $G = (Q, E)$, with $E = \{(q, q') \mid \exists a \in \Sigma, (q, a, q') \in \delta\}$. In the following, we often abuse notation and for instance say the cycle rank of A, referring to the cycle rank of its underlying graph.

There is a strong connection between the star height of a regular language, and the cycle rank of the NFAs accepting it, as witnessed by the following theorem. Theorem 2(1) is known as Eggan's Theorem [7] and proved in its present form by Cohen [6]. Theorem 2(3) is due to McNaughton [21].

Theorem 2. *For any regular language L,*

1. $sh(L) = \min\{cr(A) \mid A \text{ is an NFA accepting } L\}$. *[7, 6].*
2. $sh(L) \cdot |\Sigma| \geq \min\{cr(A) \mid A \text{ is a non-returning state-labeled NFA accepting } L\}$.
3. *if L is bideterministic, then $sh(L) = cr(A)$, where A is the minimal trim DFA accepting L. [21]*

3 Succinctness w.r.t. NFAs

In this section, we study the complexity of translating extended regular expressions into NFAs. We show that such a translation can be done in exponential time, by constructing the NFA by induction on the structure of the expression.

Proposition 1. *Let r be a RE(&, ∩, #). An NFA A with at most $2^{|r|}$ states, such that $L(r) = L(A)$, can be constructed in time $2^{\mathcal{O}(|r|)}$.*

This exponential size increase can not be avoided for any of the classes. For RE(#) this is witnessed by the expression $a^{[2^n,2^n]}$ and for RE(&) by the expression $a_1 \,\&\, \cdots \,\&\, a_n$, as already observed by Kilpelainen and Tuhkanen [18] and Mayer and Stockmeyer [20], respectively. For RE(∩), a $2^{\Omega(\sqrt{n})}$ lower bound has already been reported in [25]. The present tighter statement, however, will follow from Theorem 4 and the fact that any NFA with n states can be translated into a DFA with 2^n states [31].

Proposition 2. *For any $n \in \mathbb{N}$, there exist an RE(#) $r^{\#}$, an RE(∩) $r^{\cap}$, and an RE(&) $r^{\&}$, each of size $\mathcal{O}(n)$, such that any NFA accepting $r^{\#}$, $r^{\cap}$, or $r^{\&}$ contains at least 2^n states.*

4 Succinctness w.r.t. DFAs

In this section, we study the complexity of translating extended regular expressions into DFAs. First, from Proposition 1 and the fact that any NFA with n states can be translated into a DFA with 2^n states in exponential time [31], we can immediately conclude the following.

Proposition 3. *Let r be a RE(&, ∩, #). A DFA A with at most $2^{2^{|r|}}$ states, such that $L(r) = L(A)$, can be constructed in time $2^{2^{\mathcal{O}(|r|)}}$.*

We show that, for each of the classes RE(#), RE(∩), or RE(&), this double exponential size increase can not be avoided. For RE(#), this is witnessed by the expression $(a + b)^* a (a + b)^{[2^n,2^n]}$ which is of size $\mathcal{O}(n)$, but for which any DFA accepting it must contain at least 2^{2^n} states.

Theorem 3. *For any $n \in \mathbb{N}$ there exists an RE(#) r_n of size $\mathcal{O}(n)$ such that any DFA accepting $L(r_n)$ contains at least 2^{2^n} states.*

We now move to regular expressions extended with the intersection operator. The succinctness of RE(∩) with respect to DFAs can be obtained along the same lines as the simulation of exponential space turing machines by RE(∩) in [10].

Theorem 4. *For any $n \in \mathbb{N}$ there exists an RE(∩) $r_n^{\cap}$ of size $\mathcal{O}(n)$ such that any DFA accepting $L(r_n^{\cap})$ contains at least 2^{2^n} states.*

Proof. Let $n \in \mathbb{N}$. We start by describing the language $\mathcal{G}_n$ which will be used to establish the lower bound. This will be a variation of the following language over the alphabet $\{a, b\}$: $\{ww \mid |w| = 2^n\}$. It is well known that this language is hard to describe by a DFA. However, to define it very succinct by an RE(∩), we need to add some additional information to it.

Thereto, we first define a *marked number* as a string over the alphabet $\{0, 1, \bar{0}, \bar{1}\}$ defined by the regular expression $(0+1)^*\bar{1}\bar{0}^* + \bar{0}^*$, i.e., a binary number in which the rightmost 1 and all following 0's are marked. Then, for any $i \in [0, 2^n - 1]$ let $\text{enc}(i)$ denote the n-bit marked number encoding i. These marked numbers were introduced by Fürer in [10], where the following is observed: if $i, j \in [0, 2^n - 1]$ are such that $j = i + 1 (\text{mod } 2^n)$, then the bits of i and j which are different are exactly the marked bits of j. For instance, for $n = 2$, $\text{enc}(1) = 0\bar{1}$ and $\text{enc}(2) = \bar{1}\bar{0}$ and they differ in both bits as both bits of enc(2) are marked. Further, let $\text{enc}^R(i)$ denote the reversal of $\text{enc}(i)$.

Now, for a string $w = a_0 a_1 \dots a_{2^n-1}$ define

$$\text{enc}(w) = \text{enc}^R(0)a_0\text{enc}(0)\$\text{enc}^R(1)a_1\text{enc}(1)\$ \cdots \text{enc}^R(2^n - 1)a_{2^n-1}\text{enc}(2^n - 1)$$

and, finally, define

$$\mathcal{G}_n = \{\#\text{enc}(w)\#\text{enc}(w) \mid w \in L((a + b)^*) \wedge |w| = 2^n\}$$

For instance, for $n = 2$, and $w = abba$, $\text{enc}(w) = \bar{0}\bar{0}a\bar{0}\bar{0}\$\bar{1}0b0\bar{1}\$\bar{0}\bar{1}b\bar{1}\bar{0}\$\bar{1}1a1\bar{1}$ and hence $\#\bar{0}\bar{0}a\bar{0}\bar{0}\$\bar{1}0b0\bar{1}\$\bar{0}\bar{1}b\bar{1}\bar{0}\$\bar{1}1a1\bar{1}\#\bar{0}\bar{0}a\bar{0}\bar{0}\$\bar{1}0b0\bar{1}\$\bar{0}\bar{1}b\bar{1}\bar{0}\$\bar{1}1a1\bar{1} \in \mathcal{G}_2$.

It can be shown that any DFA accepting $\overline{\mathcal{G}_n}$, the complement of $\mathcal{G}_n$, must contain at least 2^{2^n} states. Furthermore, we can construct an expression $r_n^{\cap}$ of size $\mathcal{O}(n)$ defining $\overline{\mathcal{G}_n}$. Here, $r_n^{\cap}$ is the disjunction of many expressions, each describing some mistake a string can make in order not to be in $\mathcal{G}_n$. □

We can now extend the results for RE(∩) to RE(&). We do this by using a technique of Mayer and Stockmeyer [20] which allows, in some sense, to simulate an RE(∩) by an RE(&). To formally define this, we need some notation. Let $w = a_0 \cdots a_n$ be a string over an alphabet Σ, and let c be a symbol not in Σ. Then, for any $i \in \mathbb{N}$, define $\text{pump}_i(w) = a_0^i c^i a_1^i c^i \cdots a_k^i c^i$. Now, they proved the following:

Lemma 1 ([20]). *Let r be an RE(∩) containing k ∩-operators. Then, there exists an RE(&) s of size at most $|r|^2$ such that for any $w \in \Sigma^*$, $w \in L(r)$ iff $pump_k(w) \in L(s)$.*

That is, the expression s constructed in this lemma may define additional strings, but the set of valid pumped string it defines, corresponds exactly to $L(r)$. Using this lemma, we can now prove the following theorem.

Theorem 5. *For any $n \in \mathbb{N}$ there exists an RE(&) $r_n^{\&}$ of size $\mathcal{O}(n^2)$ such that any DFA accepting $L(r_n^{\&})$ contains at least 2^{2^n} states.*

5 Succinctness w.r.t. Regular Expressions

In this section, we study the translation of extended regular expressions to (standard) regular expressions. First, for the class RE(#) it has already been shown by Kilpelainen and Tuhkanen [18] that this translation can be done in exponential time, and that an exponential size increase can not be avoided. Furthermore, from Proposition 1 and the fact that any NFA with n states can be translated into a regular expression in time $2^{\mathcal{O}(n)}$ [9] it immediately follows that:

Proposition 4. *Let r be a RE(&, ∩, #). A regular expression s equivalent to r can be constructed in time $2^{2^{\mathcal{O}(|r|)}}$*

Furthermore, from the results in [12] (see also Figure 1(b)) it follows that in a translation from RE(∩) to standard regular expressions, a double exponential size increase can not be avoided.

Hence, it only remains to show a double exponential lower bound on the translation from RE(&) to standard regular expressions, which is exactly what we will do in the rest of this section. Thereto, we proceed in several steps and define several families of languages. First, we introduce the family of languages $(\mathcal{K}_n)_{n\in\mathbb{N}}$, on which all following languages will be based, and establish its star height. The star height of languages will be our tool for proving lower bounds on the size of regular expressions defining these languages. Then, we define the family $(\mathcal{L}_n)_{n\in\mathbb{N}}$ which is a binary encoding of $(\mathcal{K}_n)_{n\in\mathbb{N}}$ and show that these languages can be defined as the intersection of small regular expressions.

Finally, we define the family $(\mathcal{M}_n)_{n\in\mathbb{N}}$ which is obtained by simulating the intersection of the previously obtained regular expressions by the interleaving of related expressions, similar to the simulation of RE(∩) by RE(&) in Section 4. Bringing everything together, this then leads to the desired result: a double exponential lower bound on the translation of RE(&) to RE.

As an intermediate corollary of this proof, we also obtain a double exponential lower bound on the translation of RE(∩) to RE, similar to a result in [12]. We note, however, that the succinctness results for RE(&) can not be obtained by using the results in [12], and that, hence, the different lemmas which prove the succinctness of RE(∩) are necessary to obtain the subsequent results on RE(&).

5.1 $\mathcal{K}_n$: The Basic Language

We first introduce the family $(\mathcal{K}_n)_{n\in\mathbb{N}}$ defined by Ehrenfeucht and Zeiger over an alphabet whose size grows quadratically with the parameter n [8]:

Definition 1 *Let $n \in \mathbb{N}$ and $\Sigma_n = \{a_{i,j} \mid 0 \leq i, j \leq n-1\}$. Then, $\mathcal{K}_n$ contains exactly all strings of the form $a_{0,i_1}a_{i_1,i_2}\cdots a_{i_k,n-1}$ where $k \in \mathbb{N} \cup \{0\}$.*

An alternative definition of $\mathcal{K}_n$ is through the minimal DFA accepting it. Thereto, let $A_n^{\mathcal{K}} = (Q, q_0, \delta, F)$ be defined as $Q = \{q_0, \ldots, q_{n-1}\}$, $F = \{q_{n-1}\}$, and for all $i, j \in [0, n-1], (q_i, a_{i,j}, q_j) \in \delta$. That is, $A_n^{\mathcal{K}}$ is the complete DFA on n states where the transition from state i to j is labeled by $a_{i,j}$.

We now determine the star height of $\mathcal{K}_n$. This is done by observing that $\mathcal{K}_n$ is bideterministic, such that, by Theorem 2(3), $\mathrm{sh}(\mathcal{K}_n) = \mathrm{cr}(A_n^{\mathcal{K}})$, and subsequently showing that $\mathrm{cr}(A_n^{\mathcal{K}}) = n$.

Lemma 2. *For any $n \in \mathbb{N}$, $sh(\mathcal{K}_n) = n$.*

5.2 $\mathcal{L}_n$: Succinctness of RE(∩)

In this section we want to construct a set of small regular expressions such that any expression defining their intersection must be large (that is, of double exponential size). Ideally, we would like to use the family of languages $(\mathcal{K}_n)_{n\in\mathbb{N}}$ for this as we have shown in the previous section that they have a large star height, and thus by Theorem 1 can not be defined by small expressions. Unfortunately, this is not possible as the alphabet of $(\mathcal{K}_n)_{n\in\mathbb{N}}$ grows quadratically with n.

Therefore, we will introduce in this section the family of languages $(\mathcal{L}_n)_{n\in\mathbb{N}}$ which is a binary encoding of $(\mathcal{K}_n)_{n\in\mathbb{N}}$ over a fixed alphabet. Thereto, let $n \in \mathbb{N}$ and recall that $\mathcal{K}_n$ is defined over the alphabet $\Sigma_n = \{a_{i,j} \mid i, j \in [0, n-1]\}$. Now, for $a_{i,j} \in \Sigma_n$, define the function ρ_n as

$$\rho_n(a_{i,j}) = \#\mathrm{enc}(j)\$\mathrm{enc}(i)\triangle\mathrm{enc}(i+1)\triangle\cdots\triangle\mathrm{enc}(n-1)\triangle,$$

where $\mathrm{enc}(k)$, for $k \in \mathbb{N}$, denotes the $\lceil\log(n)\rceil$-bit marked number encoding k as defined in the proof of Theorem 4. So, the encoding starts by the encoding of the second index, followed by an ascending sequence of encodings of all numbers from the first index to $n-1$. We extend the definition of ρ_n to strings in the usual way: $\rho_n(a_{0,i_1}\cdots a_{i_{k-1},n-1}) = \rho_n(a_{0,i_1})\cdots\rho_n(a_{i_k,n-1})$.

We are now ready to define $\mathcal{L}_n$.

Definition 2 *Let $\Sigma = \{0, 1, \bar{0}, \bar{1}, \$, \#, \triangle\}$. For $n \in \mathbb{N}$, $\mathcal{L}_n = \{\rho_n(w) \mid w \in \mathcal{K}_n\}$.*

For instance, for $n = 3$, $a_{0,1}a_{1,2} \in \mathcal{K}_3$ and hence $\rho_3(a_{0,1}a_{1,2}) = \#0\bar{1}\$\bar{0}\bar{0}\triangle 0\bar{1}\triangle\bar{1}\bar{0}\triangle$ $\#\bar{1}\bar{0}\$0\bar{1}\triangle\bar{1}\bar{0}\triangle \in \mathcal{L}_3$. We now show that this encoding does not affect the star height. This is done by observing that, due to the specific encoding of $\mathcal{K}_n$, $\mathcal{L}_n$ is still bideterministic. Then, we obtain the star height of $\mathcal{L}_n$ by determining the cycle rank of the minimal DFA accepting it.

Lemma 3. *For any $n \in \mathbb{N}$, $sh(\mathcal{L}_n) = n$.*

Further, it can be shown that $\mathcal{L}_n$ can be described as the intersection of a set of small regular expressions.

Lemma 4. *For every $n \in \mathbb{N}$, there are regular expressions $r_1, \ldots, r_m$, with $m = 4n + 3$, each of size $\mathcal{O}(n)$, such that $\bigcap_{i\leq m} L(r_i) = \mathcal{L}_{2^n}$.*

Although it is not our main interest, we can now obtain the following by combining Theorem 1, and Lemmas 3 and 4.

Corollary 1. *For any $n \in \mathbb{N}$, there exists an RE(∩) r of size $\mathcal{O}(n^2)$ such that any (standard) regular expression defining $L(r)$ is of size at least $2^{\frac{1}{3}(2^n-1)} - 1$.*

5.3 $\mathcal{M}_n$: Succinctness of RE(&)

In this section we will finally show that RE(&) are double exponentially more succinct than standard regular expressions. We do this by simulating the intersection of the regular expressions obtained in the previous section, by the interleaving of related expressions, similar to the simulation of RE(∩) by RE(&) in Section 4. This approach will partly yield the following family of languages. For any $n \in \mathbb{N}$, define

$$\mathcal{M}_n = \{\text{pump}_{4\lceil \log n \rceil + 3}(w) \mid w \in \mathcal{L}_n\}$$

As $\mathcal{M}_n$ is very similar to $\mathcal{L}_n$, we can easily extend the result on the star height of $\mathcal{L}_n$ (Lemma 3) to $\mathcal{M}_n$:

Lemma 5. *For any $n \in \mathbb{N}$, $sh(\mathcal{M}_n) = n$.*

However, the language we will eventually define will not be exactly $\mathcal{M}_n$. Therefore, we need an additional lemma, for which we first introduce some notation. For $k \in \mathbb{N}$, and an alphabet Σ, we define $\Sigma^{(k)}$ to be the language defined by the expression $(\bigcup_{\sigma \in \Sigma} \sigma^k)^*$, i.e., all strings which consist of a sequence of blocks of identical symbols of length k. Further, for a language L, define $\text{index}(L) = \max\{i \mid i \in \mathbb{N} \wedge \exists w, w' \in \Sigma^*, a \in \Sigma \text{ such that } wa^i w' \in L\}$. Notice that $\text{index}(L)$ can be infinite. However, we will only be interested in languages for which it is finite, as in the following lemma.

Lemma 6. *Let L be a regular language, and $k \in \mathbb{N}$, such that index$(L) \leq k$. Then, $sh(L) \cdot |\Sigma| \geq sh(L \cap \Sigma^{(k)})$.*

This lemma is proved by combining Theorem 2(2) with an algorithm that transforms any non-returning state-labeled NFA A, with $\text{index}(L(A)) \leq k$, into an NFA accepting $L(A) \cap \Sigma^{(k)}$, without increasing its cycle rank.

Now, we are finally ready to prove the desired theorem:

Theorem 6. *For every $n \in \mathbb{N}$, there are regular expressions $s_1, \ldots, s_m$, with $m = 4n+3$, each of size $\mathcal{O}(n)$, such that any regular expression defining $L(s_1) \,\&\, L(s_2) \,\&\, \cdots \,\&\, L(s_m)$ is of size at least $2^{\frac{1}{24}(2^n - 8)} - 1$.*

Proof. Let $n \in \mathbb{N}$, and let $r_1, \ldots, r_m$, with $m = 4n+3$, be the regular expressions obtained in Lemma 4 such that $\bigcap_{i \leq m} L(r_i) = \mathcal{L}_{2^n}$.

Now, it is shown in [20], that given $r_1, \ldots, r_m$, it is possible to construct regular expressions $s_1, \ldots, s_m$ such that (1) for all $i \in [1, m]$, $|s_i| \leq 2|r_i|$, and if we define $\mathcal{N}_{2^n} = L(s_1) \,\&\, \cdots \,\&\, L(s_m)$, then (2) $\text{index}(\mathcal{N}_{2^n}) \leq m$, and (3) for every $w \in \Sigma^*$, $w \in \bigcap_{i \leq m} L(r_i)$ iff $\text{pump}_m(w) \in \mathcal{N}_{2^n}$. Furthermore, it follows immediately from the construction in [20] that any string in $\mathcal{N}_{2^n} \cap \Sigma^{(m)}$ is of the form $a_1^m c^m a_2^m c^m \cdots a_l^m c^m$, i.e., $\text{pump}_m(w)$ for some $w \in \Sigma^*$.

Since $\bigcap_{i \leq m} L(r_i) = \mathcal{L}_{2^n}$, and $\mathcal{M}_{2^n} = \{\text{pump}_m(w) \mid w \in \mathcal{L}_{2^n}\}$, it hence follows that $\mathcal{M}_{2^n} = \mathcal{N}_{2^n} \cap \Sigma^{(m)}$. As furthermore, by Lemma 5, $\text{sh}(\mathcal{M}_{2^n}) = 2^n$ and $\text{index}(\mathcal{N}_{2^n}) \leq m$, it follows from Lemma 6 that $\text{sh}(\mathcal{N}_{2^n}) \geq \frac{\text{sh}(\mathcal{M}_{2^n})}{|\Sigma|} = \frac{2^n}{8}$.

So, $\mathcal{N}_{2^n}$ can be described by the interleaving of the expressions s_1 to s_m, each of size $\mathcal{O}(n)$, but any regular expression defining $\mathcal{N}_{2^n}$ must, by Theorem 1, be of size at least $2^{\frac{1}{24}(2^n-8)} - 1$. This completes the proof. □

Corollary 2. *For any $n \in \mathbb{N}$, there exists an RE(&) r_n of size $\mathcal{O}(n^2)$ such that any regular expression defining $L(r_n)$ must be of size at least $2^{\frac{1}{24}(2^n-8)} - 1$.*

This completes our paper. As a final remark, we note that all lower bounds in this paper make use of a constant size alphabet and can furthermore easily be extended to a 2-letter alphabet. For any language over an alphabet $\Sigma = \{a_1, \ldots, a_k\}$, we obtain a new language by replacing, for any $i \in [1, k]$, every symbol a_i by $b^i c^{k-i+1}$. Obviously, the size of a regular expression for this new language is at most $k+1$ times the size of the original expression, and the lower bounds on the number of states of DFAs trivially carry over. Furthermore, it is shown in [22] that this transformation does not affect the star height, and hence the lower bounds on the sizes of the regular expression also carry over.

Acknowledgement. I thank Frank Neven and the anonymous referees for helpful suggestions, and Hermann Gruber for informing me about their results on the succinctness of the interleaving operator.

References

1. G. Bex, W. Gelade, F. Neven, and S. Vansummeren. Learning deterministic regular expressions for the inference of schemas from XML data. In *WWW*, pages 825–834, 2008.
2. G. Bex, F. Neven, T. Schwentick, and K. Tuyls. Inference of concise DTDs from XML data. In *VLDB*, pages 115–126, 2006.
3. G. Bex, F. Neven, and S. Vansummeren. Inferring XML Schema Definitions from XML data. In *VLDB*, pages 998–1009, 2007.
4. A. Brüggemann-Klein. Regular expressions into finite automata. *Theoretical Computer Science*, 120(2):197–213, 1993.
5. J. Clark and M. Murata. *RELAX NG Specification*. OASIS, December 2001.
6. R. S. Cohen. Rank-non-increasing transformations on transition graphs. *Information and Control*, 20(2):93–113, 1972.
7. L. C. Eggan. Transition graphs and the star height of regular events. *Michigan Mathematical Journal*, 10:385–397, 1963.
8. A. Ehrenfeucht and H. Zeiger. Complexity measures for regular expressions. *Journal of Computer and System Sciences*, 12(2):134–146, 1976.
9. K. Ellul, B. Krawetz, J. Shallit, and M. Wang. Regular expressions: New results and open problems. *Journal of Automata, Languages and Combinatorics*, 10(4):407–437, 2005.
10. M. Fürer. The complexity of the inequivalence problem for regular expressions with intersection. In *ICALP*, pages 234–245, 1980.
11. W. Gelade, W. Martens, and F. Neven. Optimizing schema languages for XML: Numerical constraints and interleaving. In T. Schwentick and D. Suciu, editors, *ICDT*, volume 4353 of *LNCS*, pages 269–283. Springer, 2007.

12. W. Gelade and F. Neven. Succinctness of the complement and intersection of regular expressions. In *STACS*, pages 325–336, 2008.
13. H. Gruber and M. Holzer. Finite automata, digraph connectivity, and regular expression size. In *ICALP*, 2008. To Appear.
14. H. Gruber and M. Holzer. Language operations with regular expressions of polynomial size. In *DCFS*, 2008. To Appear.
15. H. Gruber and J. Johannsen. Optimal lower bounds on regular expression size using communication complexity. In Roberto M. Amadio, editor, *FoSSaCS*, volume 4962 of *LNCS*, pages 273–286. Springer, 2008.
16. A. Hume. A tale of two greps. *Software, Practice and Experience*, 18(11):1063–1072, 1988.
17. T. Jiang and B. Ravikumar. A note on the space complexity of some decision problems for finite automata. *Information Processing Letters*, 40(1):25–31, 1991.
18. P. Kilpeläinen and R. Tuhkanen. Regular expressions with numerical occurrence indicators — preliminary results. In *SPLST 2003*, pages 163–173, 2003.
19. O. Kupferman and S. Zuhovitzky. An improved algorithm for the membership problem for extended regular expressions. In K. Diks and W. Rytter, editors, *MFCS*, volume 2420 of *LNCS*, pages 446–458. Springer, 2002.
20. A. J. Mayer and L. J. Stockmeyer. Word problems-this time with interleaving. *Information and Computation*, 115(2):293–311, 1994.
21. R. McNaughton. The loop complexity of pure-group events. *Information and Control*, 11(1/2):167–176, 1967.
22. R. McNaughton. The loop complexity of regular events. *Information Sciences*, 1(3):305–328, 1969.
23. A. R. Meyer and L. J. Stockmeyer. The equivalence problem for regular expressions with squaring requires exponential space. In *FOCS*, pages 125–129, 1972.
24. H. Petersen. Decision problems for generalized regular expressions. In *DCAGRS*, pages 22–29, 2000.
25. H. Petersen. The membership problem for regular expressions with intersection is complete in LOGCFL. In Helmut Alt and Afonso Ferreira, editors, *STACS*, volume 2285 of *LNCS*, pages 513–522. Springer, 2002.
26. J. M. Robson. The emptiness of complement problem for semi extended regular expressions requires c^n space. *Information Processing Letters*, 9(5):220–222, 1979.
27. R. Schott and J. C. Spehner. Shuffle of words and araucaria trees. *Fundamenta Informatica*, 74(4):579–601, 2006.
28. C.M. Sperberg-McQueen and H. Thompson. XML Schema. http://www.w3.org/XML/Schema, 2005.
29. L. J. Stockmeyer and A. R. Meyer. Word problems requiring exponential time: Preliminary report. In *STOC*, pages 1–9, 1973.
30. L. Wall, T. Christiansen, and J. Orwant. *Programming Perl*. O'Reilly, third edition, 2000.
31. S. Yu. Regular languages. In G. Rozenberg and A. Salomaa, editors, *Handbook of formal languages*, volume 1, chapter 2, pages 41–110. Springer, 1997.
32. S. Yu. State complexity of regular languages. *Journal of Automata, Languages and Combinatorics*, 6(2):221–234, 2001.

Appendix

For the convenience of the reader we give the full proofs of those theorems which only have partial proofs. The point where the proof in the main text ended is indicated by $\boxed{\cdots}$.

Before giving the proofs, we introduce an additional lemma. Thereto, we say that a graph H is a *minor* of a graph G if H can be obtained from G by removing edges, removing vertices, and contracting edges. Here, contracting an edge between two vertices u and v, means replacing u and v by one new vertex, which inherits all incoming and outgoing edges of u and v.

Lemma 7. *If a graph H is a minor of a graph G, then $cr(H) \leq cr(G)$.*

Proof. It suffices to show that if H is obtained from G by applying only one operation, then $\mathrm{cr}(H) \leq \mathrm{cr}(G)$, as the statement then remains to hold after applying multiple operations. For the operations deleting a vertex or deleting an edge, observe that H is a subgraph of G. Now, it has been shown by Cohen [6] that in this case $\mathrm{cr}(H) \leq \mathrm{cr}(G)$. For the case of an edge contraction, we apply an old result of McNaughton [21]. He shows that if there exists a so-called *pathwise homomorphism* from G to H, then $\mathrm{cr}(H) \leq \mathrm{cr}(G)$. Without going into detail, we note that such a pathwise homomorphism from G to H can easily be constructed if H is obtained from G by contracting an edge. □

Proofs for Section 2

Proof of Theorem 2(2): *For any regular language L,*

$$sh(L) \cdot |\Sigma| \geq \min\{cr(A) \mid A \text{ is a non-returning state-labeled NFA accepting } L\}$$

Proof. Let L be a regular language over an alphabet Σ. By Eggan's Theorem (Theorem 2(1)), we know that there exists an NFA A, with $L(A) = L$ and $\mathrm{cr}(A) = \mathrm{sh}(L)$. We show that, given A, we can construct a non-returning state-labeled NFA B^{sl} equivalent to A such that $\mathrm{cr}(A) \cdot |\Sigma| \geq \mathrm{cr}(B^{\mathrm{sl}})$ from which the theorem follows.

Let $A = (Q, q_0, \delta, F)$ be an NFA over Σ, we construct B^{sl} in two steps. First, we construct a non-returning NFA $B = (Q^B, q_B, \delta^B, F^B)$, with $L(B) = L(A)$, as follows: $Q^B = Q \cup \{q_B\}$, $\delta^B = \delta \cup \{(q_B, a, q) \mid q \in Q, a \in \Sigma, (q_0, a, q) \in \delta\}$, and $F^B = F$ if $q_0 \notin F$, and $F^B = F \cup \{q_B\}$, otherwise. Intuitively, B is A extended with a new initial state which only inherits the outgoing transitions of the old initial state. It should be clear that B is non-returning and $L(B) = L(A)$. Furthermore, $\mathrm{cr}(B) = \mathrm{cr}(A)$ because q_B has no incoming transitions and it thus forms a separate strongly connected component in B whose cycle rank is 0. From the definitions it then follows that $\mathrm{cr}(B) = \mathrm{cr}(A)$.

From B, we now construct the non-returning state-labeled NFA B^{sl} such that $\mathrm{cr}(B^{\mathrm{sl}}) \leq \mathrm{cr}(B) \cdot |\Sigma| = \mathrm{cr}(A) \cdot |\Sigma|$. Let $B^{\mathrm{sl}} = (Q^{\mathrm{sl}}, q_0^{\mathrm{sl}}, \delta^{\mathrm{sl}}, F^{\mathrm{sl}})$ be defined as

– $Q^{\mathrm{sl}} = \{q^a \mid q \in Q^B, a \in \Sigma\}$;
– $q_0^{\mathrm{sl}} = q_B^a$, for some $a \in \Sigma$;
– $\delta^{\mathrm{sl}} = \{(q^a, b, p^b) \mid q, p \in Q^B, a, b \in \Sigma, (q, b, p) \in \delta^B\}$; and
– $F^{\mathrm{sl}} = \{q^a \mid q \in F^B, a \in \Sigma\}$

That is, B^{sl} contains $|\Sigma|$ copies of every state q of B, each of which captures all incoming transitions of q for one alphabet symbol. Obviously, B^{sl} is a non-returning state-labeled NFA with $L(B) = L(B^{\mathrm{sl}})$.

We conclude by showing that $\mathrm{cr}(B) \cdot |\Sigma| \geq \mathrm{cr}(B^{\mathrm{sl}})$. In the following, we abuse notation and, for a set of states P, write $B[P]$ for the subautomaton of B induced by P, defined in the obvious way. Now, for a set of states P of B, let $P^{\mathrm{sl}} = \{q^a \mid q \in P, a \in \Sigma\}$, and observe that $(B[Q \setminus P])^{\mathrm{sl}} = B^{\mathrm{sl}}[Q^{\mathrm{sl}} \setminus P^{\mathrm{sl}}]$ always holds. We now show $\mathrm{cr}(B) \cdot |\Sigma| \geq \mathrm{cr}(B^{\mathrm{sl}})$ by induction on the number of states of B. If $|Q^B| = 1$ then either the single state does not contain a loop, such that $\mathrm{cr}(B) = \mathrm{cr}(B^{\mathrm{sl}}) = 0$, or the single state contains a self loop, in which case $\mathrm{cr}(B) = 1$, and as $|Q^{\mathrm{sl}}| = |\Sigma|$, $\mathrm{cr}(B^{\mathrm{sl}}) \leq |\Sigma|$ holds, and hence $\mathrm{cr}(B) \cdot |\Sigma| \geq \mathrm{cr}(B^{\mathrm{sl}})$.

For the induction step, if B is acyclic, then, again, $\mathrm{cr}(B) = \mathrm{cr}(B^{\mathrm{sl}}) = 0$. Otherwise, if B is strongly connected, then $\mathrm{cr}(B) = \mathrm{cr}(B[Q \setminus \{q\}]) + 1$, for some $q \in Q^B$. Then, $\mathrm{cr}(B) \cdot |\Sigma| = \mathrm{cr}(B[Q \setminus \{q\}]) \cdot |\Sigma| + |\Sigma|$, which by the induction hypothesis gives us $\mathrm{cr}(B) \cdot |\Sigma| \geq \mathrm{cr}(B^{\mathrm{sl}}[Q^{\mathrm{sl}} \setminus \{q\}^{\mathrm{sl}}]) + |\Sigma|$. Finally, it is shown in [13] that removing a set of states P from a graph can never decrease its cycle rank by more than $|P|$. Therefore, as $|\{q\}^{\mathrm{sl}}| = |\Sigma|$, $\mathrm{cr}(B^{\mathrm{sl}}[Q \setminus \{q\}^{\mathrm{sl}}]) + |\Sigma| \geq \mathrm{cr}(B^{\mathrm{sl}})$, and hence $\mathrm{cr}(B) \cdot |\Sigma| \geq \mathrm{cr}(B^{\mathrm{sl}})$.

Otherwise, if B consists of several strongly connected components $Q_1, \ldots, Q_k$, with $k \geq 2$, then $\mathrm{cr}(B) = \max_{i \leq k} \mathrm{cr}(B[Q_i])$. But now, by construction, every strongly connected component V of B^{sl} is contained in Q_i^{sl}, for some $i \in [1, k]$. Then, $B^{\mathrm{sl}}[V]$ is a minor of $B^{\mathrm{sl}}[Q_i^{\mathrm{sl}}]$, and hence it follows from Lemma 7 that $\mathrm{cr}(B^{\mathrm{sl}}) = \max_{V \text{ SCC of } B^{\mathrm{sl}}} \mathrm{cr}(B^{\mathrm{sl}}[V]) \leq \max_{i \leq k} \mathrm{cr}(B^{\mathrm{sl}}[Q_i^{\mathrm{sl}}])$. Now, by induction, $\mathrm{cr}(B[Q_i]) \cdot |\Sigma| \geq \mathrm{cr}(B^{\mathrm{sl}}[Q_i^{\mathrm{sl}}])$ for all $i \in [1, k]$, from which it follows that $\mathrm{cr}(B) \cdot |\Sigma| \geq \mathrm{cr}(B^{\mathrm{sl}})$.

Proofs for Section 3

Proof of Proposition 1: *Let r be a RE(&, ∩, #). An NFA A with at most $2^{|r|}$ states, such that $L(r) = L(A)$, can be constructed in time $2^{\mathcal{O}(|r|)}$.*

Proof. We construct A by induction on the structure of the formula. For the base cases, $r = \varepsilon$, $r = \emptyset$, and $r = a$, for $a \in \Sigma$, and the induction cases $r = r_1 r_2$, $r = r_1 + r_2$, and r_1^* this can easily be done using standard constructions. We give the full construction for the three special operators:

– If $r = r_1 \cap r_2$, for $i \in [1, 2]$, let $A_i = (Q^i, q_0^i, \delta^i, F^i)$ accept $L(r_i)$. Then, $A = (Q, q_0, \delta, F)$ is defined as $Q = Q_1 \times Q_2$, $q_0 = (q_0^1, q_0^2)$, $F = F_1 \times F_2$, and $\delta = \{((q_1, q_2), a, (p_1, p_2)) \mid (q_1, a, p_1) \in \delta_1 \wedge (q_2, a, p_2) \in \delta_2\}$.

– If $r = r_1 \,\&\, r_2$, then A is defined exactly as for $r_1 \cap r_2$, except for δ which now equals $\{((q_1, q_2), a, (p_1, q_2)) \mid (q_1, a, p_1) \in \delta_1\} \cup \{((q_1, q_2), a, (q_1, p_2)) \mid (q_2, a, p_2) \in \delta_2\}$.
– If $r = r_1^{[k,\ell]}$, let A_1 accept $L(r_1)$. Then, let B_1 to B_ℓ be ℓ identical copies of A_1 with disjoint sets of states. For $i \in [1, \ell]$, let $B_i = (Q^i, q_0^i, \delta^i, F^i)$. Now, define $A = (Q, q_0^0, \delta, F)$ accepting $r_1^{[k,\ell]}$ as follows: $Q = \bigcup_{i \leq \ell} Q^i$, $F = \bigcup_{k \leq i \leq \ell} F^i$, and $\delta = \bigcup_{i \leq \ell} \delta^i \cup \{(q_i, a, q_0^{i+1}) \mid q_i \in Q^i \wedge \exists p_i \in F^i \text{ such that } (q_i, a, p_i) \in \delta_i\}$.

We note that the construction for the interleaving operator comes from [20], where it was already used for a translation from RE(&) to NFAs. We argue that A contains at most $2^{|r|}$ states. For $r = r_1 \cap r_2$ or $r = r_1 \,\&\, r_2$, by induction A_1 and A_2 contain at most $2^{|r_1|}$ and $2^{|r_2|}$ states and, hence, A contains at most $2^{|r_1|} \cdot 2^{|r_2|} = 2^{|r_1|+|r_2|} \leq 2^{|r|}$ states. For $r = r_1^{[k,\ell]}$, similarly, A contains at most $\ell \cdot 2^{|r_1|} = 2^{|r_1|+\log \ell} \leq 2^{|r|}$ states. Furthermore, as the intermediate automata never have more than $2^{|r|}$ states and we have to do at most $|r|$ such constructions, the total construction can be done in time $2^{\mathcal{O}(|r|)}$.

Proofs for Section 4

Proof of Theorem 3: *For any $n \in \mathbb{N}$ there exists an RE(#) r_n of size $\mathcal{O}(n)$ such that any DFA accepting $L(r_n)$ contains at least 2^{2^n} states.*

Proof. Let $n \in \mathbb{N}$ and define $r_n = (a+b)^* a (a+b)^{[2^n, 2^n]}$. Here, r_n is of size $\mathcal{O}(n)$ since the integers in the numerical predicate are stored in binary. We show that any DFA $A = (Q, q_0, \delta, F)$ accepting $L(r_n)$ has at least 2^{2^n} states. Towards a contradiction, suppose that A has less than 2^{2^n} states and consider all strings of length 2^n containing only a's and b's. As there are exactly 2^{2^n} such strings, and A contains less than 2^{2^n} states, there must be two different such strings w, w' and a state q of A such that both $(q_0, w, q) \in \delta^*$ and $(q_0, w', q) \in \delta^*$. But now, as $w \neq w'$, there exists some $i \in [1, 2^n]$ such that the ith position of w contains an a, and the ith position of w' contains a b (or the other way around, but that is identical). Therefore, $wa^i \in L(r_n)$, and $w'a^i \notin L(r_n)$ but wa^i and $w'a^i$ are either both accepted or both not accepted by A, and hence $L(r_n) \neq L(A)$, which gives us the desired contradiction.

Proof of Theorem 4: *For any $n \in \mathbb{N}$ there exists an RE(∩) $r_n^\cap$ of size $\mathcal{O}(n)$ such that any DFA accepting $L(r_n^\cap)$ contains at least 2^{2^n} states.*

Proof. Let $n \in \mathbb{N}$. We start by describing the language $\mathcal{G}_n$ which will be used to establish the lower bound. This will be a variation of the following language over the alphabet $\{a, b\}$: $\{ww \mid |w| = 2^n\}$. It is well known that this language is hard to describe by a DFA. However, to define it very succinct by an RE(∩), we need to add some additional information to it.

Thereto, we first define a *marked number* as a string over the alphabet $\{0, 1, \bar{0}, \bar{1}\}$ defined by the regular expression $(0+1)^*\bar{1}\bar{0}^* + \bar{0}^*$, i.e., a binary number in which the rightmost 1 and all following 0's are marked. Then, for any $i \in [0, 2^n - 1]$ let $\text{enc}(i)$ denote the n-bit marked number encoding i. These marked numbers were introduced in [10], where the following is observed: if $i, j \in [0, 2^n - 1]$ are such that $j = i + 1 (\text{mod } 2^n)$, then the bits of i and j which are different are exactly the marked bits of j. For instance, for $n = 2$, $\text{enc}(1) = 0\bar{1}$ and $\text{enc}(2) = \bar{1}\bar{0}$ and they differ in both bits as both bits of enc(2) are marked. Further, let $\text{enc}^R(i)$ denote the reversal of $\text{enc}(i)$.

Now, for a string $w = a_0 a_1 \dots a_{2^n-1}$ define

$$\text{enc}(w) = \text{enc}^R(0)a_0\text{enc}(0)\$\text{enc}^R(1)a_1\text{enc}(1)\$ \cdots \text{enc}^R(2^n - 1)a_{2^n-1}\text{enc}(2^n - 1)$$

and, finally, define

$$\mathcal{G}_n = \{\#\text{enc}(w)\#\text{enc}(w) \mid w \in L((a + b)^*) \wedge |w| = 2^n\}$$

For instance, for $n = 2$, and $w = abba$, $\text{enc}(w) = \bar{0}\bar{0}a\bar{0}\bar{0}\$\bar{1}0b0\bar{1}\$\bar{0}\bar{1}b\bar{1}\bar{0}\$\bar{1}1a1\bar{1}$ and hence $\#\bar{0}\bar{0}a\bar{0}\bar{0}\$\bar{1}0b0\bar{1}\$\bar{0}\bar{1}b\bar{1}\bar{0}\$\bar{1}1a1\bar{1}\#\bar{0}\bar{0}a\bar{0}\bar{0}\$\bar{1}0b0\bar{1}\$\bar{0}\bar{1}b\bar{1}\bar{0}\$\bar{1}1a1\bar{1} \in \mathcal{G}_2$.

$\boxed{\cdots}$ Now, lets consider $\overline{\mathcal{G}_n}$, the complement of $\mathcal{G}_n$. Using a standard argument, similar to the in one in the proof of Theorem 3, it is straightforward to show that any DFA accepting $\overline{\mathcal{G}_n}$ must contain at least 2^{2^n} states. We conclude by showing that we can construct a regular expression $r_n^\cap$ of size $\mathcal{O}(n)$ defining $\overline{\mathcal{G}_n}$.

Note that $\Sigma = \{0, \bar{0}, 1, \bar{1}, a, b, \$, \#\}$. We define $N = \{0, \bar{0}, 1, \bar{1}\}$, $S = \{a, b\}$, $D = \{\$, \#\}$ and for any set U, and $\sigma \in U$, let $U_\sigma = U \setminus \{\sigma\}$. Now, we construct a set of expressions, each capturing a possible mistake in a string. Then, $r_n^\cap$ simply is the disjunction of these expressions. The expressions are as follows.
All strings which do not start with #:

$$\varepsilon + \Sigma_\# \Sigma^*$$

All strings in which two symbols at a distance $n + 1$ do not match:

$$\Sigma^*(N\Sigma^n(S + D) + S\Sigma^n(S + N) + D\Sigma^n(D + N))\Sigma^*$$

All strings which do not end with $\text{enc}(2^n - 1) = 1^{n-1}\bar{1}$:

$$\Sigma^*(\Sigma_{\bar{1}} + \Sigma_1 \Sigma^{[1,n-1]})$$

All strings in which # occurs before any other number than $\text{enc}^R(0)$ or where \$ occurs before $\text{enc}^R(0)$:

$$\Sigma^*(\$\bar{0}^n + \#(\Sigma^{[0,n-1]}\Sigma_{\bar{0}}))\Sigma^*$$

All strings which contain more or less than 2 #-symbols

$$\Sigma_\#^*(\# + \varepsilon)\Sigma_\#^* + \Sigma^*\#\Sigma^*\#\Sigma^*\#\Sigma^*$$

All strings which contain a (non-reversed) binary number which is not correctly marked:

$$\Sigma^* S N^*((0+1)(\bar{0}+\$+\#)+(\bar{0}+\bar{1})N_{\bar{0}})\Sigma^*$$

All strings in which the binary encodings of two numbers surrounding an a or b are not each others reverse. Thereto, we first define expressions r_i, for all $i \in [0,n]$, such that $L(r_i) = \{w\sigma w' \mid \sigma \in S \wedge w, w' \in \Sigma^* \wedge |w| = |w'| \wedge |w| \leq i\}$, inductively as follows: $r_0 = S$ and for all $j \in [1,n]$, $r_j = r_0 + \Sigma r_{j-1} \Sigma$. Then, the following is the desired expression:

$$\Sigma^*(r_n \cap \bigcup_{\sigma \in N} \sigma \Sigma^* \Sigma_\sigma)\Sigma^*$$

All strings in which the binary encodings of two numbers surrounding a \$ or # do not differ by exactly one, i.e., there is a substring of the form $\mathrm{enc}(i)\$\mathrm{enc}^R(j)$ or $\mathrm{enc}(i)\#\mathrm{enc}^R(j)$ such that $j \neq i+1(\bmod 2^n)$. Exactly as above, we can inductively define r'_n such that $L(r'_n) = \{w\sigma w' \mid \sigma \in D \wedge w, w' \in \Sigma^* \wedge |w| = |w'| \wedge |w| \leq n\}$. Then, we obtain:

$$\Sigma^*(r'_n \cap ((0+\bar{0})\Sigma^*(\bar{0}+1)+(1+\bar{1})\Sigma^*(\bar{1}+0)))\Sigma^*$$

All strings in which two a or b symbols which should be equal are not equal. We now define expressions s_i, for all $i \in [0,n]$ such that $L(s_i) = \{wu\#vw^R \mid u,v,w \in \Sigma^* \wedge |w| = i\}$. By induction, $s_0 = \Sigma^*\#\Sigma^*$, and for all $i \in [1,n]$, $s_i = \Sigma s_{i-1} \Sigma \cap \bigcup_{\sigma \in \Sigma} \sigma \Sigma^* \sigma$. Then, the following is the desired expression:

$$\Sigma^*(a s_n b + b s_n a)\Sigma^*$$

Now, a string is not in $\mathcal{G}_n$ iff it is accepted by at least one of the previous expressions. Hence, $r_n^\cap$, defined as the disjunction of all these expressions, defines exactly $\overline{\mathcal{G}_n}$. Furthermore, notice that all expressions, including the inductively defined ones, are of size $\mathcal{O}(n)$, and hence $r_n^\cap$ is also of size $\mathcal{O}(n)$. This concludes the proof.

Proof of Theorem 5: *For any $n \in \mathbb{N}$ there exists an RE(&) $r_n^\&$ of size $\mathcal{O}(n^2)$ such that any DFA accepting $L(r_n^\&)$ contains at least 2^{2^n} states.*

Proof. Let $n \in \mathbb{N}$ and consider the expression $r_n^\cap$ of size $\mathcal{O}(n)$ constructed in Theorem 4 such that any DFA accepting $L(r_n^\cap)$ contains at least 2^{2^n} states. Now, let $r_n^\&$ be the regular expression *simulating* $r_n^\cap$ obtained from Lemma 1, such that for some $k \in \mathbb{N}$ and any $w \in \Sigma^*$, $w \in L(r_n^\cap)$ iff $\mathrm{pump}_k(w) \in L(r_n^\&)$. Then, $r_n^\&$ is of size $\mathcal{O}(n^2)$ and, exactly as before, it is straightforward to show that any DFA accepting $L(r_n^\&)$ contains at least 2^{2^n} states.

Proofs and Figures for Section 5

Proof of Lemma 2: *For any $n \in \mathbb{N}$, $sh(\mathcal{K}_n) = n$.*

Proof. We start by observing that, for any $n \in \mathbb{N}$, the language $\mathcal{K}_n$ is bideterministic. Indeed, the inverse of the DFA $A_n^{\mathcal{K}}$ accepting $\mathcal{K}_n$ is again deterministic as every transition is labeled with a different symbol. Furthermore, $A_n^{\mathcal{K}}$ is the minimal trim DFA accepting $\mathcal{K}_n$. Hence, by Theorem 2(3), $\mathrm{sh}(\mathcal{K}_n) = \mathrm{cr}(A_n^{\mathcal{K}})$. We conclude by showing that, for any $n \in \mathbb{N}$, $\mathrm{cr}(A_n^{\mathcal{K}}) = n$.

Thereto, first observe that the graph underlying $A_n^{\mathcal{K}}$ is the complete graph (including self-loops) on n nodes, which we denote by K_n. We proceed by induction on n. For $n = 1$, the graph is a single node with a self loop and hence by definition $\mathrm{cr}(K_1) = 1$. For the inductive step, suppose that $\mathrm{cr}(K_n) = n$ and consider K_{n+1} with node set V_{n+1}. Since V_{n+1} consists of only one strongly connected component, $\mathrm{cr}(K_{n+1}) = 1 + \min_{v \in V_{n+1}}\{\mathrm{cr}(K_{n+1}[V_{n+1} \setminus \{v\}])\}$. However, for any $v \in V_{n+1}$, $K_{n+1}[V_{n+1} \setminus \{v\}] = K_n$, and hence by the induction hypothesis $\mathrm{cr}(K_{n+1}) = n + 1$.

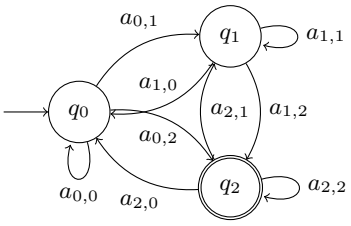

Fig. 2. The DFA $A_3^{\mathcal{K}}$, accepting $\mathcal{K}_3$.

Proof of Lemma 3: *For any $n \in \mathbb{N}$, $sh(\mathcal{L}_n) = n$.*

Proof. Let $n \in \mathbb{N}$. We first show that $\mathrm{sh}(\mathcal{L}_n) \leq n$. By Lemma 2, $\mathrm{sh}(\mathcal{K}_n) = n$, and hence there exists a regular expression $r_{\mathcal{K}}$, with $L(r_{\mathcal{K}}) = \mathcal{K}_n$ and $\mathrm{sh}(r_{\mathcal{K}}) = n$. Let $r_{\mathcal{L}}$ be the regular expression obtained from $r_{\mathcal{K}}$ by replacing every symbol $a_{i,j}$ by $\rho_n(a_{i,j})$. Obviously, $L(r_{\mathcal{L}}) = \mathcal{L}_n$ and $\mathrm{sh}(r_{\mathcal{L}}) = n$, and hence $\mathrm{sh}(\mathcal{L}_n) \leq n$.

We now show that $\mathrm{sh}(\mathcal{L}_n) \geq n$. The proof is along the same lines as the proof of Lemma 2. We first show that $\mathcal{L}_n$ is bideterministic and can then determine its star height by looking at the minimal DFA accepting it. In fact, the reason for the slightly involved encoding of $\mathcal{K}_n$ is precisely the bideterminism property.

To show that $\mathcal{L}_n$ is bideterministic, we now construct the minimal DFA $A_n^{\mathcal{L}}$ accepting $\mathcal{L}_n$. Here, $A_n^{\mathcal{L}}$ will consist of n identical subautomata B_n^0 to B_n^{n-1} defined as follows. For any $i \in [0, n-1]$, $B_n^i = (Q^i, q_n^i, \delta_i, F^i)$ is the smallest automaton for which Q^i contains distinct states $q_0^i, \ldots, q_n^i$ and $p_0^i, \ldots, p_{n-1}^i$ such that for any $j \in [0, n-1]$,

– $(q_j^i, \mathrm{enc}(j)\triangle, q_{j+1}^i) \in \delta_i^*$, and for all $w \neq \mathrm{enc}(j)\triangle$, $(q_j^i, w, q_{j+1}^i) \notin \delta_i^*$; and
– $(q_n^i, \#\mathrm{enc}(j), p_j^i) \in \delta_i^*$, and for all $w \neq \#\mathrm{enc}(j)$, $(q_n^i, w, p_j^i) \notin \delta_i^*$.

As an example, Figure 3 shows B_3^2. Now, $A_n^{\mathcal{L}} = (Q, q_n^0, \delta, F)$ is defined as $Q = \bigcup_{i<n} Q^i$, $F = \{q_n^{n-1}\}$ and $\delta = \bigcup_{i<n} \delta_i \cup \{(p_j^i, \$, q_i^j) \mid i, j \in [0, n-1]\}$. Figure 4 shows $A_3^{\mathcal{L}}$.

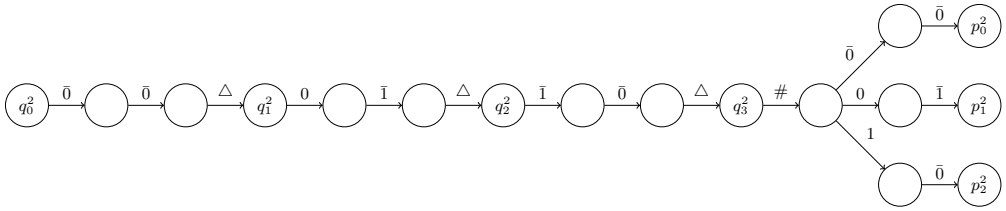

Fig. 3. The automaton B_3^2.

To see that $A_n^{\mathcal{L}}$ indeed accepts $\mathcal{L}_n$, notice that after reading a substring $\#\mathrm{enc}(i)$, for some i, the automaton moves to sub-automaton B_n^i. Then, after passing through B_n^i which ends by reading a new substring $\#\mathrm{enc}(j)$, the automaton moves to state q_i^j of sub-automaton B_n^j. This ensures that the subsequent ascending sequence of numbers starts with $\mathrm{enc}(i)$. Hence, the automaton checks correctly whether the numbers which should be equal, are equal.

Furthermore, $A_n^{\mathcal{L}}$ is bideterministic and minimal. To see that it is bideterministic, notice that all states except the q_j^i only have one incoming transition. Furthermore, the q_j^i each have exactly two incoming transitions, one labeled with \$ and one labeled with $\triangle$. Minimality then follows immediately from the fact that $A_n^{\mathcal{L}}$ is both trim and bideterministic.

Now, since $\mathcal{L}_n$ is bideterministic and $A_n^{\mathcal{L}}$ is the minimal trim DFA accepting $\mathcal{L}_n$, it follows from Theorem 2(3) that $\mathrm{sh}(\mathcal{L}_n) = \mathrm{cr}(A_n^{\mathcal{L}})$. Therefore, it suffices to show that $\mathrm{cr}(A_n^{\mathcal{L}}) \geq n$. Thereto, observe that $A_n^{\mathcal{K}}$, the minimal DFA accepting $\mathcal{K}_n$, is a minor of $A_n^{\mathcal{L}}$. Indeed, we can easily contract edges and remove nodes from $A_n^{\mathcal{K}}$ such that the only remaining nodes are q_n^0 to q_n^{n-1}, and such that they form a complete graph. Now, since it is shown in Lemma 2 that $\mathrm{cr}(A_n^{\mathcal{K}}) = n$ and by Lemma 7 and the fact that $A_n^{\mathcal{K}}$ is a minor of $A_n^{\mathcal{L}}$, we know that $\mathrm{cr}(A_n^{\mathcal{L}}) \geq \mathrm{cr}(A_n^{\mathcal{K}})$, it follows that $\mathrm{sh}(\mathcal{L}_n) = \mathrm{cr}(A_n^{\mathcal{L}}) \geq n$. This completes the proof.

Proof of Lemma 4: *For every $n \in \mathbb{N}$, there are regular expressions $r_1, \ldots, r_m$, with $m = 4n + 3$, each of size $\mathcal{O}(n)$, such that $\bigcap_{i \leq m} L(r_i) = \mathcal{L}_{2^n}$.*

Proof. Let $n \in \mathbb{N}$, and recall that $\mathcal{L}_{2^n}$ is defined over the alphabet $\Sigma = \{0, 1, \bar{0}, \bar{1}, \$, \#, \triangle\}$. Let $N = \{0, 1, \bar{0}, \bar{1}\}$, $D = \{\$, \#, \triangle\}$ and for any $\sigma \in \Sigma$, let $\Sigma_\sigma = \Sigma \setminus \{\sigma\}$. The expressions are as follows.

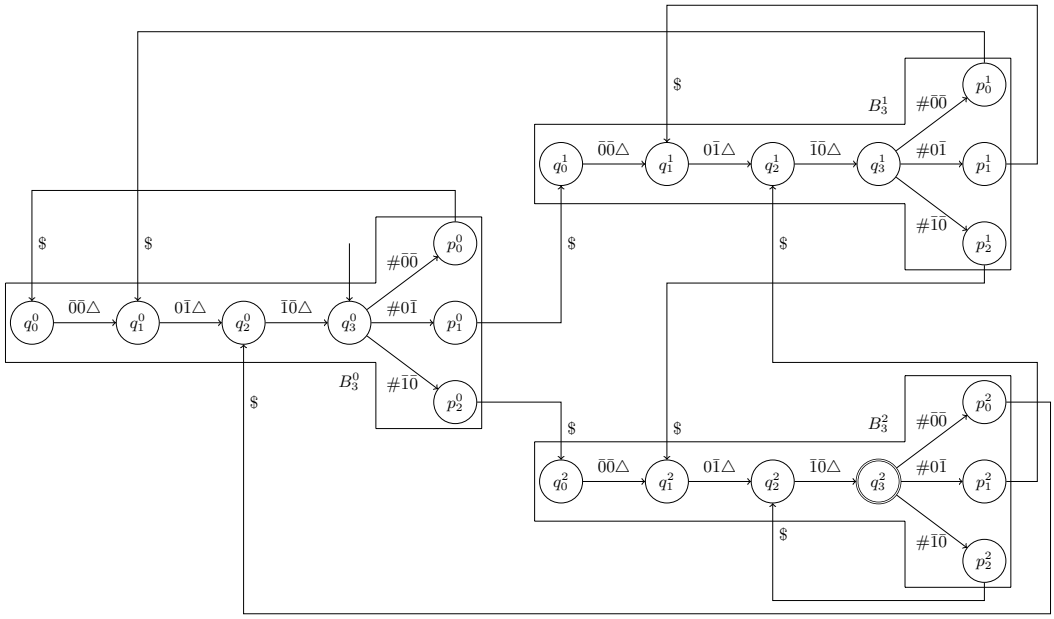

Fig. 4. The DFA $A_3^{\mathcal{L}}$, accepting $\mathcal{L}_3$.

The format of the string has to be correct:

$$(\#N^n\$N^n\triangle(N^n\triangle)^+)^+$$

Every number should be properly marked:

$$(D^+((0+1)^*\bar{1}\bar{0}^* + \bar{0}^*))^*$$

Every number before a \$ should be equal to the number following the next \$. We define two sets of regular expressions. First, for all $i \in [0, n-1]$, the $(i+1)$th bit of the number before an even \$ should be equal to the $(i+1)$th bit of the number after the next \$.

$$(\#N^i \bigcup_{\sigma \in N} (\sigma \Sigma_\$^* \$ \Sigma_\$^* \$ N^i \sigma) \Sigma_\#^*)^* (\varepsilon + \#\Sigma_\#^*)$$

Second, for all $i \in [0, n-1]$, the $(i+1)$th bit of the number before an odd \$ should be equal to the $(i+1)$th bit of the number after the next \$.

$$\#\Sigma_\#^* (\#N^i \bigcup_{\sigma \in N} (\sigma \Sigma_\$^* \$ \Sigma_\$^* \$ N^i \sigma) \Sigma_\#^*)^* (\varepsilon + \#\Sigma_\#^*)$$

Every two (marked) numbers surrounding a $\triangle$ should differ by exactly one. Again, we define two sets of regular expressions. First, for all $i \in [0, n-1]$, the

$(i+1)$th bit of the number before an even $\triangle$ should properly match the $(i+1)$th bit of the next number:

$$(\#\Sigma_{\$}^{*}((\triangle+\$)\Sigma^{i}((0+\bar{0})\Sigma_{\#}^{n}(\bar{1}+0)+(1+\bar{1})\Sigma_{\#}^{n}(\bar{0}+1))\Sigma^{n-i-1})^{*}((\triangle+\$)\Sigma^{n}+\varepsilon)\triangle)^{*}$$

Second, for all $i \in [0, n-1]$, the $(i+1)$th bit of the number before an odd $\triangle$ should properly match the $(i+1)$th bit of the next number:

$$(\#\Sigma_{\$}^{*}\$\Sigma^{n}(\triangle\Sigma^{i}((0+\bar{0})\Sigma_{\#}^{n}(\bar{1}+0)+(1+\bar{1})\Sigma_{\#}^{n}(\bar{0}+1))\Sigma^{n-i-1})^{*}(\triangle\Sigma^{n}+\varepsilon)\triangle)^{*}$$

Every ascending sequence of numbers should end with $\mathrm{enc}(2^n-1) = 1^{n-1}\bar{1}$:

$$(\#\Sigma_{\#}^{*}1^{n-1}\bar{1}\triangle)^{*}$$

Proof of Corollary 1: *For any $n \in \mathbb{N}$, there exists an RE(∩) r of size $\mathcal{O}(n^2)$ such that any (normal) regular expression defining $L(r)$ is of size at least $2^{\frac{1}{3}(2^n-1)}-1$.*

Proof. Let $n \in \mathbb{N}$. By Lemma 4 there exists a linear number of regular expressions of linear size, such that their intersection defines $\mathcal{L}_{2^n}$. Hence, there also exists an RE(∩) r of size $\mathcal{O}(n^2)$ defining $\mathcal{L}_{2^n}$. Furthermore, by Lemma 3, $\mathrm{sh}(\mathcal{L}_{2^n}) = 2^n$ and therefore by Theorem 1 any regular expression defining $\mathcal{L}_{2^n}$ is of size at least $2^{\frac{1}{3}(2^n-1)} - 1$.

Proof of Lemma 5: *For any $n \in \mathbb{N}$, $sh(\mathcal{M}_n) = n$.*

Proof. The proof is along the same lines as the proof of Lemma 3. Again, $\mathcal{M}_n$ is bideterministic as witnessed by the minimal DFA $A_n^{\mathcal{M}}$ accepting $\mathcal{M}_n$. In fact, $A_n^{\mathcal{M}}$ is simply obtained from $A_n^{\mathcal{L}}$ by replacing each transition over a symbol a by a sequence of states reading $a^m c^m$, with $m = 4\lceil \log n \rceil + 3$. Some care has to be taken, however, for the states q_j^i, as these have two incoming transitions: one labeled with \$ and one labeled with $\triangle$. Naively creating these two sequences of states leading to q_j^i, we obtain a non-minimal DFA. However, if we merge the last m states of these two sequences (the parts reading c^m), we obtain the minimal trim DFA $A_n^{\mathcal{M}}$ accepting $\mathcal{M}_n$. Then, the proof proceeds exactly as the proof of Lemma 3.

Proof of Lemma 6: *Let L be a regular language, and $k \in \mathbb{N}$, such that index$(L) \leq k$. Then, $sh(L) \cdot |\Sigma| \geq sh(L \cap \Sigma^{(k)})$.*

Proof. Let L be a regular language, and $k \in \mathbb{N}$, such that $\mathrm{index}(L) \leq k$. We show that, given any non-returning state-labeled NFA A accepting L, we can can construct an NFA B such that $L(B) = L \cap \Sigma^{(k)}$ and B is a minor of A.

We show how this implies the lemma. Let A be any non-returning state-labeled NFA of minimal cycle rank accepting L, and let B be as constructed above. First, since B is a minor of A it follows from Lemma 7 that $\mathrm{cr}(B) \leq$

$\mathrm{cr}(A)$. Furthermore, by Eggan's Theorem (Theorem 2(1)), $\mathrm{cr}(B) \geq \mathrm{sh}(L(B)) = \mathrm{sh}(L \cap \Sigma^{(k)})$, and hence $\mathrm{cr}(A) \geq \mathrm{sh}(L \cap \Sigma^{(k)})$. Now, since A is a non-returning state-labeled NFA of minimal cycle rank, we can conclude from Theorem 2(2) that $\mathrm{sh}(L) \cdot |\Sigma| \geq \mathrm{cr}(A)$, and thus $\mathrm{sh}(L) \cdot |\Sigma| \geq \mathrm{sh}(L \cap \Sigma^{(k)})$.

We conclude by giving the construction of B. Thereto, let $A = (Q, q_0, \delta, F)$ be a non-returning state-labeled NFA. Now, for any $q \in Q$, and $a \in \Sigma$, define the function $\mathrm{in}_a(q)$ as follows:

$$\mathrm{in}_a(q) = \max\{i \mid i \in \mathbb{N} \cup \{0\} \wedge \exists w \in \Sigma^* \text{ such that } (q_0, wa^i, q) \in \delta^*\}$$

Notice that as i can equal zero and $\mathrm{index}(L)$ is finite, $\mathrm{in}_a(q)$ is well defined for any state which is not useless. Intuitively, $\mathrm{in}_a(q)$ represents the maximal number of a symbols which can be read when entering state q.

Now, the following algorithm transforms A, accepting L, into B, accepting $L(A) \cap \Sigma^{(k)}$. Repeat the following two steps, until no more changes are made:

1. Apply one of the following rules, if possible:
   (a) If exists $q, q' \in Q$, $a \in \Sigma$, with $(q, a, q') \in \delta$, such that $\mathrm{in}_a(q') > \mathrm{in}_a(q)+1$ $\Rightarrow$ remove (q, a, q') from δ.
   (b) If exists $q, q' \in Q$, $a \in \Sigma$, with $(q, a, q') \in \delta$ and $q \neq q_0$, such that $\mathrm{in}_{\mathrm{symbol}(q)}(q) < k$ and $\mathrm{symbol}(q) \neq a \Rightarrow$ remove (q, a, q') from δ.
   (c) If exists $q \in F$, $q \neq q_0$, such that $\mathrm{in}_{\mathrm{symbol}(q)}(q) < k \Rightarrow$ make q non-final, i.e., remove q from F.
2. Remove all useless states from A, and recompute $\mathrm{in}_a(q)$ for all $q \in Q$, $a \in \Sigma$.

It remains to show that B, the automaton obtained when no more rules can be applied, is the desired automaton. That is, that B is a minor of A and that $L(B) = L(A) \cap \Sigma^{(k)}$. It is immediate that B is a minor of A since B is obtained from A by only removing transitions or states.

To show that $L(B) = L(A) \cap \Sigma^{(k)}$, we first proof that $L(A) \cap \Sigma^{(k)} \subseteq L(B)$. Thereto, let $A_1, \ldots, A_n$, with $A_1 = A$ and $A_n = B$, be the sequence of NFAs produced by the algorithm, where each A_i is obtained from A_{i-1} by applying exactly one rule and possibly removing useless states. It suffices to show that for all $i \in [1, n-1]$, $L(A_i) \cap \Sigma^{(k)} \subseteq L(A_{i+1}) \cap \Sigma^{(k)}$, as $L(A) \cap \Sigma^{(k)} \subseteq L(B) \cap \Sigma^{(k)} \subseteq L(B)$ then easily follows.

Before proving $L(A_i) \cap \Sigma^{(k)} \subseteq L(A_{i+1}) \cap \Sigma^{(k)}$, we introduce some notation. For any $q \in Q$, $a \in \Sigma$, we define

$$\mathrm{out}_a(q) = \max\{i \mid i \in \mathbb{N} \cup \{0\} \wedge \exists w \in \Sigma^*, q_f \in F \text{ such that } (q, a^i w, q_f) \in \delta^*\}$$

Here, $\mathrm{out}_a(q)$ is similar to $\mathrm{in}_a(q)$ and represents the maximal number of a's which can be read when leaving q. Since of course $L(A_i) \subseteq L(A)$ holds for all $i \in [1, n]$ it also holds that $\mathrm{index}(A_i) \leq k$. Therefore, for any A_i, any state q of A_i and any $a \in \Sigma$, it holds that

$$\mathrm{in}_a(q) + \mathrm{out}_a(q) \leq k \tag{1}$$

We are now ready to show that $L(A_i) \cap \Sigma^{(k)} \subseteq L(A_{i+1}) \cap \Sigma^{(k)}$. Thereto, we prove that for any $w \in L(A_i) \cap \Sigma^{(k)}$, any accepting run of A_i on w is still an

accepting run of A_{i+1} on w. More precisely, we prove for every rule separately that if A_{i+1} is obtained from A_i by applying this rule, then the assumption that the removed state or transition is used in an accepting run of A_i on w leads to a contradiction.

For rule (1a), suppose transition (q, a, q') is removed but (q, a, q') occurs in an accepting run of A_i on w, i.e., $w = uav$ for $u, v \in \Sigma^*$, $(q_0, u, q) \in \delta^*$ and $(q', v, q_f) \in \delta^*$, for some $q_f \in F$. Since $w \in \Sigma^{(k)}$, there exists a $j \in [0, n-1]$ and $u', v' \in \Sigma^*$ such that $u = u'a^j$ and $v = a^{k-j-1}v'$. It immediately follows that $\text{in}_a(q) \geq j$, and $\text{out}_a(q') \geq k - j - 1$. Then, since $\text{in}_a(q') > \text{in}_a(q) + 1$, also $\text{in}_a(q') + \text{out}_a(q') > j + k - j - 1 + 1 = k$. However, this contradicts equation (1).

For rules (1b) en (1c), we describe the obtained contradictions less formal. For rule (1b), if the transition (q, a, q'), with $\text{symbol}(q) \neq a$ is used in an accepting run on w, then q is entered after reading k $\text{symbol}(q)$ symbols, and hence $\text{in}_{\text{symbol}(q)}(q) \geq k$. Contradiction. For rule (1c), again, if q is the accepting state of some run on w, then q must be preceded by k $\text{symbol}(q)$ symbols, and hence $\text{in}_{\text{symbol}(q)}(q) \geq k$, contradiction.

We now show $L(B) \subseteq L(A) \cap \Sigma^{(k)}$. Thereto, let $B = (Q^B, q_0^B, \delta_B, F^B)$ and first consider the following observation. Let (q, a, q') be a transition in δ_B. Since B does not contain useless states, it is easy to see that $\text{in}_a(q') \geq \text{in}_a(q) + 1$. As rule (1a) can not be applied to transition (q, a, q'), we now obtain the following equality:

$$\text{For } q, q' \in Q^B, a \in \Sigma, \text{ with } (q, a, q') \in \delta_B : \text{in}_a(q) + 1 = \text{in}_a(q') \qquad (2)$$

We are now ready to show that $L(B) \subseteq L(A) \cap \Sigma^{(k)}$. Since $L(B) \subseteq L(A)$ definitely holds, it suffices to show that $L(B) = L(B) \cap \Sigma^{(k)}$, i.e., B only accepts strings in $\Sigma^{(k)}$.

Towards a contradiction, suppose that B accepts a string $w \notin \Sigma^{(k)}$. Then, there must exist $i \in [1, k-1]$, $a \in \Sigma$, $u \in L(\varepsilon + \Sigma^*(\Sigma \setminus \{a\}))$ and $v \in L(\varepsilon + (\Sigma \setminus \{a\})\Sigma^*)$, such that $w = ua^iv$. Furthermore, as $w \in L(B)$, there also exist states $p_0, \ldots, p_i$, such that $(q_0, u, p_0) \in \delta_B^*$, $(p_i, v, q_f) \in \delta_B^*$ for some $q_f \in F$, and $(p_i, a, p_{i+1}) \in \delta_B$ for all $i \in [0, i-1]$.

We first argue that $\text{in}_a(p_i) = k$. By equation (1) it suffices to show that $\text{in}_a(p_i) \geq k$. Notice that, as $i > 0$, $\text{symbol}(p_i) = a$. We consider two cases. If $v = \varepsilon$, then $p_i \in F$ and hence by rule (1c) $\text{in}_a(p_i) \geq k$. Otherwise, $v = bv'$ for $b \in \Sigma$, $b \neq a$ and $v' \in \Sigma^*$ and hence p_i must have an outgoing b-labeled transition, with $b \neq a = \text{symbol}(p_i)$. By rule (1b), $\text{in}_a(p_i) \geq k$ must hold.

Now, by repeatedly applying equation (2), we obtain $\text{in}_a(p_j) = k - (i - j)$, for all $j \in [0, i]$ and, in particular, $\text{in}_a(p_0) = k - i > 0$. This gives us the desired contradiction. To see why, we distinguish two cases. If $u = \varepsilon$, then $p_0 = q_0$. As A was non-returning and we didn't introduce any new transitions, B is also non-returning and hence $\text{in}_a(q_0) = 0$ should hold. Otherwise, if $u = u'b$ for $u' \in \Sigma^*$ and $b \neq a$, then p_0 has an incoming b transition. As B is state-labeled, p_0 does

not have an incoming a transition, and hence, again, $\mathrm{in}_a(p_0) = 0$ should hold. This concludes the proof.