T U M

I N S T I T U T F Ü R I N F O R M A T I K

Finite Automata, Digraph Connectivity, and Regular Expression Size

Hermann Gruber Markus Holzer

TUM-I0725
Dezember 07

T E C H N I S C H E U N I V E R S I T Ä T M Ü N C H E N

TUM-INFO-12-I0725-0/1.-FI

Druck: Institut für Informatik der
Technischen Universität München

Finite Automata, Digraph Connectivity, and Regular Expression Size

Hermann Gruber
Institut für Informatik, Ludwig-Maximilians-Universität München,
Oettingstraße 67, 80538 München, Germany
gruberh@tcs.ifi.lmu.de

Markus Holzer
Institut für Informatik, Technische Universität München,
Boltzmannstraße 3, D-85748 Garching, Germany
holzer@in.tum.de

Abstract

We study some descriptional complexity aspects of regular expressions. In particular, we give lower bounds on the minimum required size for the conversion of deterministic finite automata into regular expressions and on the required size of regular expressions resulting from applying some basic language operations on them, namely intersection, shuffle, and complement. Some of the lower bounds obtained are asymptotically tight, and, notably, the examples we present are over an alphabet of size two. To this end, we develop a new lower bound technique that is based on the star height of regular languages. It is known that for a restricted class of regular languages, the star height can be determined from the digraph underlying the transition structure of the minimal finite automaton accepting that language. In this way, star height is tied to cycle rank, a structural complexity measure for digraphs proposed by Eggan and Büchi, which measures the degree of connectivity of directed graphs. This measure, although a robust graph theoretic notion, appears to have remained widely unnoticed outside formal language theory. We establish some basic theory for cycle rank and put it into context with other digraph connectivity measures proposed recently.

1 Introduction

One of the most basic theorems in formal language theory is that every finite automaton can be effectively converted into an equivalent regular expression, and *vice versa* [24]. While algorithms accomplishing these tasks have been known for a long time, there has been a renewed interest in these classical problems during the last few years. For instance, new algorithms for converting regular expressions into finite automata outperforming classical algorithms have been found only recently, as well as a matching lower bound of $\Omega(n \cdot \log^2 n)$ on the minimum number

of transitions required by any equivalent nondeterministic finite automaton (NFA). The lower bound is, however, only reachable for growing alphabets, and a better algorithm is known for constant alphabet size, see [36] for the current state of the art.

In contrast, much less is known about the converse direction, namely of converting finite automata into regular expressions. Apart from the fundamental nature of the problem, some applications of converting finite automata into regular expressions lie in control flow normalization, including uses in software engineering such as automatic translation of legacy code [29]. All known algorithms covering the general case of infinite languages are based on the classical ones, which are compared in the survey [35]. The drawback is that all of these (structurally similar) algorithms return expressions of size $2^{O(n)}$ in the worst case, and Ehrenfeucht and Zeiger exhibit a family of languages over a growing alphabet of size n^2 for which this exponential blow-up is inevitable [10]. These examples naturally raise the question whether such a blow-up can also occur for constant alphabet size, a question posed in [12]. One of the main results in this paper gives a positive answer to this question, even in the case of a binary alphabet. In contrast, for the case of unary languages, it is known that every n-state nondeterministic finite automaton can be converted in polynomial time into an equivalent regular expression of polynomial size [25].

Currently, there are not many lower bound techniques for regular expression size besides those based on nondeterministic state complexity. A notable exception is the technique used in the above mentioned work [10], which however appears to require a largely growing alphabet, cf. [12]. It is not hard to see that a straightforward approach of using a standard binary encoding that was suggested in [39] fails, since the languages encoded in this way admit polynomial-sized regular expressions. A different technique, based on communication complexity, was proposed recently in [15]. This technique gives an optimal super-polynomial lower bound for the conversion problem in the case of finite languages, even for alphabet size two. However, it is unclear if and how this technique could be possibly generalized to infinite languages, in order to yield a truly exponential lower bound.

In this paper, we take a different direction that relates the minimum regular expression size to the (restricted) star height of regular languages. The star height is a structural complexity measure of regular languages that has been intensively studied in the literature for more than 40 years, see [23] for a recent treatment of this topic. The latter concept is related to the cycle rank of digraphs, a connectivity measure for directed graphs defined by Eggan and Büchi [9] in the 1960s. Determining the star height can be sometimes reduced to the easier task of determining the cycle rank of a certain digraph; moreover, measuring the connectivity of directed graphs is a very active research topic in the current literature, see, e.g., [5, 4, 22, 31]. Since we feel that this measure is of interest in its own right, we summarize and further develop some parts of the theory of cycle rank and compare it to some of the measures recently proposed in the literature.

These connections turn out to be fruitful, and we are able not only to prove a lower bound of $2^{\Omega(n)}$ on the problem of converting a given n-state deterministic finite automaton over a binary alphabet into an equivalent regular expression, but also to give reasonably good lower bounds for the alphabetic width of some basic regular language operations, namely intersection, complement and shuffle. The first three of the four results partially answer respective open questions posed in [12]; in particular, the first of these is settled completely in this work.

The paper is organized as follows: In the next section, we fix some basic definitions regarding

(di)graphs and formal languages. Thereafter, we investigate the cycle rank of directed graphs in detail. The discussion starts with connections between cycle rank and the star height of regular languages. We continue with lifting some basic bricks of the theory of undirected graphs to the directed case. Then we give a characterization of cycle rank in terms of some cops and robbers game. Finally, we put this measure into context with other digraph connectivity measures that have been proposed recently in the literature. In the last section, we use some of these properties and relations to deduce lower bounds for regular expression size, in particular a lower bound for the alphabetic width of the intersection of two regular expressions that is exponential in the size of the smaller expression, a roughly doubly-exponential lower bound for the complement operation, and an exponential lower bound for the shuffle operation, similar to the one given for intersection. Interestingly, these bounds are achievable for languages over a binary alphabet that are structurally extremely simple. Finally, we give a lower bound of $2^{\Omega(n)}$ on the required size for converting deterministic finite automata into regular expressions, even for binary alphabet.

For the convenience of the reader, we have starred some sections that contain background material which is not essential for proving lower bounds on regular expression size. These can be omitted in a first reading, but are recommended to gain a deeper understanding of the topic, especially for readers with a bias towards graph theory.

2 Basic Definitions

2.1 Digraphs and Graphs

We fix some basic notations from graph theory. A *directed graph*, or *digraph*, $G = (V, E)$ consists of a finite set of vertices V with an associated set of edges $E \in V \times V$. For an edge $e = (u, v) \in E$, we refer to v as the *head* of e, denoted by $\mathrm{head}(e)$ and to v as the tail of e, denoted by $\mathrm{tail}(e)$. We say an edge e *enters* (*leaves*, respectively) a vertex v if $\mathrm{head}(e) = v$ (if $\mathrm{tail}(e) = v$, respectively). For a vertex v, the *indegree* of v is the number of edges entering v, and is denoted by $\mathrm{indeg}(v)$. Similar, its outdegree is number of edges leaving v, and is denoted by $\mathrm{outdeg}(v)$. An edge whose head and tail are identical is called a *loop*. If G has no loops, then G is called *loop-free*.

If the edge relation of G is symmetric, then G is an *undirected graph*, or simply *graph*. It is often convenient to view the set of edges of an undirected graph as a set of unordered pairs $\{u, v\}$, with u and v in V. Only if there is no risk of confusion, for an undirected graph G, we refer to the set $\{\, \{u, v\} \mid (u, v) \in E \,\}$ as the set of edges of G, and, abusing notation, denote it by E.

Taking the symmetric closure of the edge relation of a digraph G results in an undirected graph, called the undirected graph of G, denoted by $\overleftrightarrow{G}$. Intuitively, this graph is obtained by forgetting the orientation of edges in G.

A digraph $H = (U, F)$ is a *subdigraph*, or simply *subgraph*, of a digraph $G = (V, E)$, if $U \subseteq V$ and for each edge $(u, v) \in F$ with $u, v \in U$, the pair (u, v) is an edge in E. A subgraph H is called *induced* if furthermore for each edge $(u, v) \in E$ with $u, v \in U$, the pair (u, v) is also an edge in E. In the latter case, H is referred to as the subgraph of G induced by U, and denoted by $G[U]$.

We recall the definitions of some other important concepts related to walks and reachability. A subgraph $H = (U, F)$ of G is *strongly connected*, if for every vertices u and v, both u is reachable from v and v is reachable from u. A strongly connected subgraph H is called *nontrivial* if H has at least one edge, otherwise it is called trivial. Note that a every trivial strongly connected subgraph has at most one vertex, but if G is not loop-free, it also has nontrivial strongly connected subgraphs with only one vertex. A set of vertices $\emptyset \subset C \subseteq V$ is a strongly connected component if $G[C]$ is strongly connected, but for every proper superset $C' \supset C$, the induced subgraph $G[C']$ is not strongly connected.

A digraph $G = (V, E)$ is the *directed union* of two digraphs $G_1 = (V_1, E_1)$ and $G_2 = (V_2, E_2)$ with $V_1 \cap V_2 = \emptyset$, written $G = G_1 \overrightarrow{\cup} G_2$, if $V = V_1 \cup V_2$ and $E = E_1 \cup E_2 \cup V_1 \times V_2$. If E satisfies the weaker condition $E_1 \cup E_2 \subseteq E \subseteq E_1 \cup E_2 \cup V_1 \times V_2$, then G is called a *partial* directed union of the two graphs G_1 and G_2.

2.2 Formal Languages

We introduce some basic notions in formal language and automata theory—for a thorough treatment, the reader might want to consult a textbook such as [18]. In particular, let Σ be a finite alphabet and Σ^* the set of all words over the alphabet Σ, including the empty word ε. The length of a word w is denoted by $|w|$, where $|\varepsilon| = 0$. A *(formal) language* over the alphabet Σ is a subset of Σ^*.

Apart from the usual set operations, such as union, intersection and complement with respect to Σ^*, the following operations on languages are very common: The *concatenation* of two languages L_1 and L_2, denoted by $L_1 \cdot L_2$, is defined as the set $\{ vw \mid v \in L_1 \text{ and } w \in L_2 \}$, and for $i \geq 1$, the i-th power of a language L, denoted by L^i, is defined as the i-fold concatenation of L with itself. By convention, $L^0 = \{\varepsilon\}$. The *(Kleene) star* of a language L is given by $L^* = \bigcup_{i \geq 0} L^i$.

A *nondeterministic finite automaton* (NFA) is a 5-tuple $A = (Q, \Sigma, \delta, Q_0, F)$, where Q is a finite set of states, Σ is a finite set of input symbols, $\delta : Q \times \Sigma \to 2^Q$ is the transition function, $Q_0 \in Q$ is the set of initial states, and $F \subseteq S$ is the set of accepting states. The transition function δ is extended to a function from $\delta : Q \times \Sigma^* \to 2^Q$ in the natural way, i.e., $\delta(q, \varepsilon) = \{q\}$ and $\delta(q, aw) = \bigcup_{q' \in \delta(q,a)} \delta(q', w)$, for $q \in Q$, $a \in \Sigma$, and $w \in \Sigma^*$.

A NFA *with ε-transitions* is a nondeterministic finite automaton whose transition function is extended on empty input: For each q, a subset of $P \subseteq Q$ is specified such that $\delta(q, \varepsilon) = P$. Here, we will always tacitly assume that $q \in P$, even if q does not occur explicitly in the specification of P. It is known that for every NFA with ε-transitions, there is an equivalent NFA without ε-transitions having the same number of states, see, e.g., [18]. But allowing ε-transitions can sometimes simplify some constructions on NFAs. Thus, unless stated otherwise, all NFAs used in this work can have ε-transitions.

A nondeterministic finite automaton without ε-transitions $A = (Q, \Sigma, \delta, I, F)$ is *deterministic* (a DFA), if $|I| = 1$ and $|\delta(q, a)| \leq 1$, for every $q \in Q$ and $a \in \Sigma$. In this case we simply write $\delta(q, a) = p$ instead of $\delta(q, a) = \{p\}$. The *language accepted* by a finite automaton A is

$$L(A) = \{ w \in \Sigma^* \mid \exists i \in I : \delta(i, w) \cap F \neq \emptyset \}.$$

Two (deterministic or nondeterministic) finite automata are *equivalent* if they accept the same language. A state $q \in Q$ is called *useful* if there is both a word $x \in \Sigma^*$ such that $q \in \delta(i, x)$

for some $i \in I$ and a word y such that $\delta(q, y) \cap F \neq \emptyset$. It is not hard to see that we obtain an equivalent finite automaton by restricting the transition function to the set of useful states and omitting the states from the specification which are not useful. Unless stated otherwise, we will always assume that all states in a specified automaton are useful.

A formal language L is called *regular* if and only if it is accepted by some nondeterministic finite automaton. It is well known that every regular language can indeed be also accepted by some deterministic finite automaton, so deterministic finite automata provide an alternative characterization of the class of regular languages. A third important characterization of the class of regular languages is by means of *regular expressions*:

Let Σ be an alphabet. The regular expressions over Σ and the languages that they denote are defined recursively as follows:

- $\emptyset$ is a regular expression and denotes the empty language;
- For $a \in \Sigma \cup \{\varepsilon\}$, a is a regular expression and denotes the language $\{a\}$,
- if r and s are regular expressions denoting languages R and S, then $(r + s)$, $(r \cdot s)$ and $(r)^*$ are regular expressions denoting the languages $R \cup S$, $R \cdot S$ and R^*, respectively.

The language described by the regular expression r is denoted by $L(r)$.

For convenience, parentheses are sometimes omitted and the concatenation is simply written as juxtaposition. The priority of operators is specified in the usual fashion: concatenation is performed before union, and star before both product and union.

As with finite automata, the notion of equivalence is defined based on equality of the described language. A basic theorem in the theory of regular languages states that every (non-)deterministic finite automaton can be effectively converted into an equivalent regular expression and *vice versa* [24]. So these formalisms are equally expressive, and their expressive power characterizes the regular languages.

In this work, we are primarily concerned with the economy of description of these devices when describing the same language, in particular for regular expressions. The *alphabetic width* (or *size*) $\mathrm{alph}(r)$ of a regular expression r is defined as the total number of occurrences of letters of Σ in r. For a regular language L, we define its alphabetic width, $\mathrm{alph}(L)$, as the minimum alphabetic width among all regular expressions describing L. For comparing the space requirements of regular expressions with those of finite automata, we need also a measure of descriptional complexity for (non-)deterministic finite automata. For a regular language L, the deterministic (nondeterministic, respectively) state complexity of L, denoted by $\mathrm{sc}(L)$ ($\mathrm{nsc}(L)$, respectively) is the minimal number of states needed by a DFA (NFA, respectively) accepting L.

3 Star Height of Regular Languages and Cycle Rank of Digraphs

3.1 Definitions and Early Results

For a regular expression r, the star height $h(r)$ is a structural complexity measure inductively defined by:

1. $h(r) = 0$ for $r \in \Sigma \cup \{\emptyset, \varepsilon\}$.

2. $h(r \cdot r_2) = h(r_1 + r_2) = \max(h(r_1), h(r_2))$

3. $h(r^*) = 1 + h(r)$.

The star height of a regular language L, denoted by $h(L)$, is then defined as the minimum star height among all regular expressions describing L. We will later establish a relation between star height and alphabetic width of regular languages. This relation will allow us to reduce the task of proving lower bounds on alphabetic width to the one of proving lower bounds on star height.

First, we call to attention a structural complexity measure for directed graphs (with self-loops allowed) intimately related to the star height of regular languages, called the cycle rank, suggested by Eggan and Büchi in the course of investigating the star height of regular languages [9].

Definition 1. *The* cycle rank *of a directed graph $G = (V, E)$, denoted by $cr(G)$, is inductively defined as follows:*

1. *If G is acyclic, then $cr(G) = 0$.*

2. *If G is strongly connected, then $cr(G) = 1 + \min_{v \in V}\{cr(G - v)\}$, where $G - v$ denotes the graph with the vertex set $V \setminus \{v\}$ and appropriately defined edge set.*

3. *If G is not strongly connected, then $cr(G)$ equals the maximum cycle rank among all* strongly connected component*s of G.*

In the following, we will be sometimes concerned with the cycle rank of the simple directed graph underlying the transition structure of finite automata, so for a given finite automaton A, let its cycle rank, denoted by $cr(A)$, be defined as the cycle rank of the underlying graph. The following relation between cycle rank of automata and star height of regular languages became known as Eggan's Theorem [9]:

Theorem 2 (Eggan's Theorem)**.** *The star height of a regular language L equals the minimum cycle rank among all nondeterministic finite automata accepting L.*

The star height of a regular language appears to be a more difficult concept than alphabetic width. An apparent problem is that this concept is defined as a minimum over all regular expressions describing the respective language, without bounding the expression size explicitly. The very same holds for automaton size in Eggan's reformulation. In fact, the question whether the star height of a given regular language is computable became a famous problem known as the *(restricted) star height problem*, was open for 25 years before a positive answer was given in [16], relying on heavy combinatorial machinery. For a more recent and shorter solution to this problem, the reader is referred to [23].

In light of these considerations, proving lower bounds on alphabetic width *via* lower bounds on star height appears to be trading a hard problem for an even harder one. But early research on the star height problem established a subclass of regular languages for which the star height is determined more easily.

Definition 3. *A regular language L is* bideterministic *if there exists a deterministic finite automaton with a single final state such that a deterministic finite automaton accepting the reversed language L^R is obtained by reverting the direction of all transitions and exchanging the roles of the initial and final state.*

The star height of bideterministic languages was shown to be computable in [27], building on earlier work which was, however, published only later in [28]:

Theorem 4 (McNaughton's Theorem)**.** *Let L be a bideterministic language, and let A be the minimal trim (i.e., without a dead state) deterministic finite automaton accepting L. Then $h(L) = cr(A)$.*

In fact, the minimality requirement in the above theorem is not needed, since every bideterministic finite automaton in which all states are useful is already a trim minimal DFA—in fact even a state minimal NFA [38]. Here, a state is useful if it is both reachable from the start state, and from which the final state is reachable from it.

In order to relate star height to alphabetic width, and to find lower bound techniques for the cycle rank, we study the latter concept in more detail. First, we establish a basic fact about cycle rank, which is used throughout the following sections. The second part of the following statement is found in [28, Theorem 2.4.], and the other part is established by an easy induction:

Lemma 5. *Let $G = (V, E)$ be a directed graph, and let $U \subseteq V$. Then*

$$cr(G) - |U| \leq cr(G - U) \leq cr(G),$$

where $G - U$ denotes the graph with vertex set $V \setminus U$ and appropriately defined edge set. □

3.2 Basic Properties of Cycle Rank*

In the following, we establish some basic results concerning the concept of cycle rank. Namely, we lift a part of the essential theory known for undirected graphs to the case of directed graphs.

3.2.1 Cycle Rank of Undirected Graphs*

Many connectivity measures proposed recently for digraphs coincide on symmetric digraphs with some notion already known for undirected graphs. We show that the cycle rank of a digraph can be seen as a generalization of the minimum elimination tree height $eth(G)$, a connectivity measure for undirected graphs introduced in [7], a parameter which plays an important role in sparse matrix factorization [37]. Several equivalent characterizations of this measure exist, but the one closest to the definition of cycle rank is given in [30]:

Definition 6. *Let $G = (V, E)$ be an undirected, loop-free graph. Then the elimination tree height of G, denoted by $eth(G)$, is inductively defined by:*

$$eth(G) = \begin{cases} 1 & \text{if } |V| = 1 \\ 1 + \min_{v \in V} eth(G - v) & \text{if } G \text{ is connected} \\ \max_{i=1}^{p} C_p & \text{if } C_1, C_2, \ldots, C_p \text{ connected components of } G. \end{cases}$$

Since the notions of connected component and strongly connected component coincide on undirected graphs, the following lemma is immediate from the definition of cycle rank:

Lemma 7. *Let G be a loop-free digraph. Then $cr(\overleftrightarrow{G}) = eth(G) - 1$.* □

3.2.2 Cycle Rank and Analysis Forests*

Following [27], an *analysis forest* for a digraph $G = (V, E)$ with k nontrivial strongly connected components $C_1, C_2, \ldots, C_k$ is a rooted forest $F = (\mathcal{F}, \mathcal{E})$ of k rooted trees, with edges oriented downwards towards the leaves, having the following properties:

1. Each vertex $\mathcal{F}$ is in $V \times 2^V$.
2. The roots of the k trees are of the form (x_1, C_1), (x_2, C_2),$\ldots$ (x_k, C_k).
3. There is no pair distinct vertices of the form (x, X) and (y, X) in the forest.
4. There is an edge from $(x, X) \in \mathcal{F}$ to $(y, Y) \in \mathcal{F}$ if and only if Y is a nontrivial strongly connected component in $G[X] - x$. If $((x, X), (y, Y)) \in \mathcal{E}$, we call (y, Y) a successor of (x, X).

As usual, *height* $h(T)$ of a rooted tree T is defined inductively by letting $h(T) = 0$ if T has no vertices, and otherwise $h(T)$ equals the maximum height among all subtrees whose roots are the successors of the root of T plus one. The height of a rooted forest is then defined as the maximum height among all of its rooted trees. As noted in [27], proving that the cycle rank of a digraph G equals the minimum height of among all analysis forests for G is an easy exercise. The theorem reads as follows:

Theorem 8. *Let G be a digraph, and $k \geq 0$. Then G admits an analysis forest of height k if and only if the cycle rank of G is at most k.*

Remarkably, the concept of analysis forests was recently discovered independently and studied in the context of a basic problem in linear algebra, namely LU factorization of sparse matrices [11], there called *elimination forests*. Here analysis forests arise as a natural generalization of the well-known concept of elimination trees (for symmetric matrices) introduced in [37] to the case of unsymmetric matrices.

3.2.3 Computational Complexity of Cycle Rank and a Special Case of the Star Height Problem*

The above considerations allow for a classification of the computational complexity of determining the cycle rank of directed graphs.

Theorem 9. *Given a digraph G and an integer k, deciding whether $cr(G) \leq k$ is* **NP***-complete, and this still holds when G is strongly connected.*

Proof. On the one hand, given an undirected, loop-free graph and an integer k, determining whether the minimum elimination tree height is at most k is **NP**-hard [7], and the instances appearing in the reduction are in fact strongly connected. The previous lemma shows that this measure coincides with the cycle rank for loop-free undirected graphs, thus establishing **NP**-hardness of this problem. On the other hand, an analysis forest of height at most k for a given digraph G constitutes a proof for the fact $cr(G) \leq k$. Since such a forest has at most as many nodes as there are vertices in G, this proves membership in **NP**. □

As an aside, recall from Section 3.1 that it was open for a long time whether the restricted star height of regular languages is computable at all. The precise computational complexity of this problem still remains unresolved; the best known upper bound is doubly exponential space, and the best known lower bound is **PSPACE**-hardness [23]. The first step in the proof of the mentioned upper bound is the determinization of the given finite automaton. So it can be easily observed that the upper bound drops to singly exponential space if the input is specified as a *deterministic* finite automaton. In contrast, the proof of the lower bound is by a reduction of the NFA universality problem, and hence crucially depends on the fact that the given finite automaton is nondeterministic.

The next theorem precisely determines the computational complexity for a subclass of instances, namely for bideterministic finite automata. This class was also historically the first class of finite automata for which the star height was shown to be computable [27]. Our result also shows that determining the star height remains computationally intractable if the given regular language is specified as a deterministic finite automaton.

Theorem 10. *Given a bideterministic finite automaton A and an integer k, deciding whether the star height of $L(A)$ is at most k is* **NP***-complete.*

Proof. Membership in **NP** can be seen by using Theorems 4 and 9, which together imply that determining whether the cycle rank of a given graph is at most k is in **NP**. In order to establish **NP**-hardness, we recall that we can re-interpret every strongly connected directed graph $G = (V, E)$ with two distinguished vertices s and t as a reduced bideterministic finite automaton A_G over the alphabet E accepting the set of all walks from s to t [8]. This construction clearly runs in polynomial time. Using again McNaughton's Theorem, the **NP**-hard question whether the cycle rank of G is at most k reduces to the question whether the star height of A_G is at most k. □

3.2.4 Cycle Rank and Weak Graph Separators*

Next, we lift some results concerning graph separators known for undirected graphs (see, e.g., [30]), to the case of directed graphs:

Definition 11. *Let $G = (V, E)$ be a digraph and let $U \subseteq V$ be a set of vertices. A set of vertices S is a* weak α-separator *for U if every strongly connected component of $G[U \setminus S]$ contains at most $\alpha|U|$ vertices. For real numbers $0 \leq k \leq |V|$, let $s(G, k)$ denote the maximum of the size of the smallest weak $\frac{1}{2}$-separator for U, where the maximum is taken over all subsets U of size at most k of V. The* weak separator number *of G, denoted by $s(G)$, is defined as $s(G, |V|)$.*

Then we can prove the following theorem.

Lemma 12. *Let $G = (V, E)$ be a digraph with $n \geq 1$ vertices. Then*

$$cr(G) \leq \sum_{0 \leq k \leq \log n - 1} s\left(G, \frac{n}{2^k}\right).$$

Proof. By induction on n. In the case $n = 1$, we have $s(G) = 0$, and the sum in the statement of the lemma is empty, as desired. The induction step is as follows: By definition of weak separator number, G has a weak $\frac{1}{2}$-separator S of size at most $s(G, n)$. Let $G_1, \ldots G_p$ be the

strongly connected components of $G - S$. Each of these has cardinality at most $\frac{n}{2}$. With Lemma 5, we obtain

$$cr(G) \leq |S| + \max_{1 \leq i \leq p} cr(G_i) \leq s\left(G, \frac{n}{2^0}\right) + \max_{1 \leq i \leq p} cr(G_i).$$

Since for each $k \leq n$ and for each strongly connected component C_i obviously holds $s(C_i, k) \leq s(G, k)$, we have by induction hypothesis

$$\max_{1 \leq i \leq p} cr(C_i) \leq \sum_{1 \leq k \leq \log(n/2)-1} s\left(G, \frac{n/2}{2^k}\right) = \sum_{1 \leq k \leq \log n - 1} s\left(G, \frac{n}{2^k}\right),$$

where the right hand side is obtained by simply shifting the summation index. By putting the two inequalities together, the proof is completed. □

This establishes the following relations between cycle rank and weak separator number:

Corollary 13. *Let G be a digraph with n vertices. Then $cr(G) \leq s(G) \cdot \log n$.* □

It is known that digraphs with undirected treewidth at most k have weak separator number[1] at most $k + 1$, thus we also have:

Corollary 14. *Let G be a digraph with n vertices and undirected treewidth at most k. Then $cr(G) \leq (k + 1) \cdot \log n$.* □

3.3 Cycle Rank *via* Cops and Robbers

The characterization of cycle rank in terms of some "game against the graph" was already suggested in [28]. We give a modern formulation in terms of a cops and robber game. This characterization provides a useful tool in proving lower bounds on the cycle rank of specific families of digraphs. Overmore, many other digraph connectivity measures proposed recently admit a characterization in terms of some cops and robber game; this allows to compare the cycle rank with these other measures.

The *cops and visible robber game*, defined in [4, 31], is given as follows: Let $G = (V, E)$ be a digraph. Initially, the cops occupy some set of $X \subseteq V$ vertices, with $|X| \leq k$, and the robber is placed on some vertex $v \in V - X$. At any time, some of the cops can reside outside the graph, say, in a helicopter. In each round, the cop player chooses the next location $X' \subseteq V$ for the cops. The stationary cops in $X \cap X'$ remain in their positions, while the others go to the helicopter and fly to their new position. During this, the robber player, knowing the cops' next position X' from wire-tapping the police radio, can run at great speed to any new position v', provided there is both a (possibly empty) directed path from v to v', i.e., he has to avoid to run into a stationary cop, and to run along a path in the remaining graph induced by the non-blocked vertices. Afterwards, the helicopter lands the cops at their new positions, and the next round starts, with X' and v' taking over the roles of X and v, respectively. The cop player wins the game if the robber cannot move any more, and the robber player wins if the robber can escape indefinitely.

[1] This indeed holds even for a stronger notion of separators, see [33]

The *immutable cops* variant of the above game restricts the movements of the cops in the following way: Once a cop has been placed on some vertex of the graph, he has to stay there forever. The *hot-plate* variant of the game restricts the movements of the robber in that he has to move along a nontrivial path in each move—even if the path consists only of a self-loop. The *strong* variant of the above game restricts the robbers moves in requiring that he stays in the same strongly connected of the graph $G - (X \cap X')$ induced by the non-blocked vertices.

Theorem 15. *Let G be a digraph and $k \geq 0$. Then k cops have a winning strategy for the immutable cops and hot-plate strong visible robber game if and only if the cycle rank of G is at most k, i.e., $cr(G) \geq k$.*

Proof. We proceed by induction on n, the number of vertices in G. In case $n = 1$, the graph has either no edge, or exactly one edge. Since the robber has to move in each step, no cop, respectively, one cop is required to catch the robber.

To do the induction step for the "if" part, assume G has cycle rank at most k. We will show that k cops have a winning strategy for the immutable hot-plate strong visible robber game by induction on the number n of vertices in G.

Now we consider two cases:

1. If G is not strongly connected itself, the cops play $\emptyset$ in the first move, that is, none of the cops is placed on G. Let c be the first position played by the robber, and let C be the strongly connected component containing c. Then the robber cannot ever leave the subgraph C. Thus it suffices to describe each a winning strategy for k cops on the induced subgraph $G[C]$ for every initial choice of the robber, that is, for each strongly connected component of G. Since G is not strongly connected, each strongly connected component has size at most $n - 1$, and by induction assumption, k cops have a winning strategy on each of these, and hence on G.

2. The remaining case is that G is strongly connected. In this case, by definition of cycle rank, there exists some vertex v such that $cr(G - v) = cr(G) - 1$. In this case, we place the first cop on v, and the game continues on $G - v$. Since the latter graph has $n - 1$ vertices, the induction assumption implies that the remaining $k - 1$ cops have a winning strategy on $G - v$.

To do the induction step for the "only if" part, assume k cops have a winning strategy in the above game on G. Then they also have a winning strategy which plays $\emptyset$ unless the robber has moved for the first time. That is, they have a winning strategy for every subgraph induced by some strongly connected component C of G. The cycle rank of G equals the maximum cycle rank among its strongly connected components, so if G is not strongly connected, then $cr(G) \leq k$ follows by the induction assumption. In the other case, let U be the set of vertices occupied by the cops when the cop player makes a nontrivial move for the first time. Then the game continues on $G-U$, and there is a winning strategy for $k-|U|$ cops on $G-U$. By induction assumption, we have $cr(G - U) \leq k - |U|$. By Lemma 5, we have $cr(G) \leq cr(G - U) + |U|$, and thus $cr(G) \leq k$, as desired. □

The above game is robust in the sense that small variations of rules such as letting the robber player begin, or allowing only the placement of one cop at a time does not alter the number

of required cops. Also note that at most one additional cop is needed if we drop the hot-plate restriction: The allowed movements of the robber in hot-plate variant coincide with the original ones as long as he resides in a strongly connected component of size at least two. The situation changes only when the robber is finally trapped in a strongly connected component consisting only of one vertex. If this vertex has a loop, a cop has to be placed on this vertex anyway. But if not, indeed one more cop is needed to catch the robber: Otherwise, the robber could stay on this vertex indefinitely, and thus win the game. By these considerations, we obtain the following situation on loop-free digraphs as a corollary to the above proof:

Corollary 16. *Let G be a loop-free digraph and $k \geq 0$. Then $k+1$ cops have a winning strategy for the immutable cops and strong visible robber game if and only if the cycle rank of G is at most k, i.e., $cr(G) \leq k$.* □

3.4 Relation to other Digraph Connectivity Measures

Apart from cycle rank, several other structural complexity measures for directed graphs have been proposed and studied recently, for instance directed pathwidth, entanglement, D-width, and DAG-width. Apart from entanglement, cycle rank and directed path-width, all of these measures coincide on symmetric digraphs with undirected treewidth, and the undirected counterparts of cycle rank and directed pathwidth are minimum elimination tree height and undirected pathwidth, respectively. We note that, unlike cycle rank, for quite a few of these digraph connectivity measures, the associated decision problems are not known to be in **NP**. For all above mentioned measures, a characterization in terms of cops and robbers games is known. We briefly explain the variants of the game defined in Section 3.3 that characterize the respective measures, if not already mentioned in Section 3.3:

- The *monotone* variant of the game requires that the none of the cops ever revisits a vertex that has already been occupied and vacated by some cop before.
- In the *invisible robber* variant of the game, the robber is not visible for the cops.
- The *inert robber* variant of the game is like the invisible robber variant, but the robber may only move if a cop is about to land on his current position.
- The *entanglement game* is played in the hot-plate variant, but the robber can move only along paths of length 1 in each turn. The cop players can move at most one cop at a time, and a moved cop must be on the position occupied by the robber in the previous turn.

In Table 1 we summarize the relation between digraph connectivity measures and cops and robber game variants.

For uniformity reasons, we agree here that each of the above connectivity measures—apart from the weak separator number—equals the minimum number of cops in the respective game, although some of the authors have added the constant term 1 to this number in the original definitions.

On the one hand, we show that cycle rank can be arbitrarily larger than all of these measures. A simple example is the undirected path P_n, which is clearly not acyclic. Its entanglement

Measure	cops	robber	Refs.
weak separator number	—	—	here
D-width	monotone	strong visible	[13, 34]
DAG-width	monotone	visible	[4, 31]
entanglement	entanglement game	entanglement game	[5]
Kelly-width	monotone	inert, invisible	[20]
directed path-width	monotone	invisible	[3, 19]
cycle rank	immutable	hot-plate strong visible	here

Table 1: Relation of digraph connectivity measures and cops and robber games.

equals two [5], being a symmetric graph, and indeed a path, its (un)directed path-width equals one [3], and since undirected path-width is an upper bound for undirected treewidth, its DAG-width, Kelly-width and its D-width are all equal to one. But, as observed in [28, pp. 318f.], for each $k \geq 1$, the graph P_{2^k} has cycle rank k. Hence the gap between all these measures and cycle rank can be logarithmic in the number of vertices of the digraph. It is also easy to see that the weak separator number of all undirected paths equals 1, so the bound relating weak separator number and cycle rank that we found in Lemma 12 is essentially tight. This result is not surprising, since similar results are known for undirected graphs, relating elimination tree height to pathwidth, separator number and treewidth, see [7].

On the other hand, most digraph connectivity measures are bounded above by the cycle rank: Namely, assume $m(G)$ is a digraph connectivity measure m that satisfies the following properties for every digraph G:

1. There is a global constant c such that if G is acyclic, then $m(G) = c$,
2. for each vertex v in G holds $m(G) \leq m(G - v) + 1$,
3. if $G = G_1 \overrightarrow{\cup} G_2$ is the directed union of G_1 and G_2, then

$$m(G) \leq \max\{\, m(G_1), m(G_2) \,\}.$$

Then a straightforward induction shows that $m(G) \leq cr(G) + c$.

These three properties are known to hold for most of the connectivity measures in the table above, compare [4, 20, 13]. As an interesting example, we establish these properties for the entanglement. Similar proofs can be carried out for the remaining digraph connectivity measures mentioned above.

Theorem 17. *Let G be a digraph. Then $cr(G) \geq ent(G)$.*

For the first property, the entanglement of a directed acyclic graph is zero [5], so it remains to show the remaining two properties. The following lemma establishes the second property:

Lemma 18. *Let $G = (V, E)$ be a digraph and let $U \subseteq V$. Then*

$$ent(G) - |U| \leq ent(G - U) \leq ent(G).$$

Proof. For the first inequality, a winning strategy for k cops on $G - U$ can be extended to a winning strategy for $k + |U|$ cops on G as follows: The first k detectives proceed as they would do on $G - U$. The remaining $|U|$ detectives are used to block each of the vertices in U as soon as the thief visits one of them for the first time. In this way, the first k detectives force the thief to leave the subgraph $G - U$, and the remaining $|U|$ detectives ensure that he can do this only finitely often.

For the second inequality, let $ent(G[U]) = k$. Then the thief has a certain winning strategy against $k-1$ cops in the entanglement game on $G[U]$, for $i = 1$ or 2. Then the thief can employ the very same strategy to win against k cops on G: According to this strategy, the thief never visits vertices outside the vertex set of $G[U]$, and none of the detectives can be ever placed on a vertex outside $G[U]$. In this way, the thief enforces that game is played *de facto* on $G[U]$, in the sense that no player ever visits any vertex in $V \setminus U$, and thus he wins against $k - 1$ detectives also on G. □

Finally, the behavior under directed union is determined in the following lemma:

Lemma 19. *Assume G is the directed union of G_1 and G_2. Then*

$$ent(G) = \max\{\, ent(G_1), ent(G_2) \,\}.$$

Proof. Let $k = \max\{ent(G_1), ent(G_2)\}$. Then $ent(G) \geq k$ follows immediately from the previous lemma, since both G_1 and G_2 are induced subgraphs of G. For the other inequality, namely $ent(G) \geq k$, we proceed by induction on the number n of vertices in G. For the base case $n = 2$, both G_1 and G_2 have entanglement at most 1, and the claimed inequality holds by [5, Proposition 3].

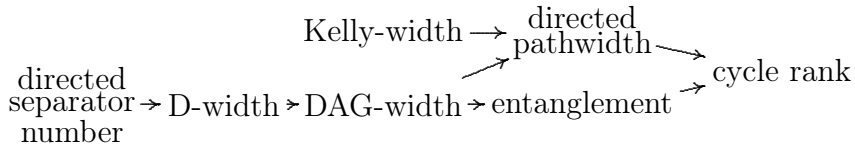
To do the induction step, assume the claim holds for all graphs G having at most $n - 1$ vertices. Initially, all detectives stay outside the graph. Let $\mathcal{C} = \{C_1, C_2, \ldots, C_r\}$, with $r \geq 1$, be the partition of V into strongly connected components, and let $\prec$ be the strict partial order on $\mathcal{C}$ induced by the reachability relation, that is, $C \prec C'$ if there is a path from some vertex in C to some vertex in C'. Since the thief has to travel along an edge in every move, he will eventually enter some maximal strongly connected component C. The strategy of the detectives is now to force the thief to eventually leave the component C if he is not to be caught. Thus the detectives enter the graph G one by one as the thief moves inside C, each occupying some vertex in C. Since G is the directed sum of two nonempty graphs, $|C| \leq n - 1$, and $G[C]$ is a proper induced subgraph of either G_1 or G_2. Thus, by Lemma 18, k cops have a winning strategy on the graph $G[C]$. In this way they can force the thief to eventually leave the component C.

Again, eventually the thief has to enter another maximal connected component C' with $C \prec C'$, $G[C']$ is again a proper induced subgraph of of either G_1 or G_2, and the detectives can proceed in a similar way they did for C, forcing the thief to leave C' again. As the strict partial order $\prec$ has no infinite chains, there is some maximal strongly connected component which the thief cannot leave any more, and in which he is finally caught. □

These properties together imply the relation between cycle rank and entanglement stated in Theorem 17.

To conclude this section, we summarize some of the known and new relations between the connectivity measures under consideration. Some of them follow immediately from their definition as cops-and-robbers game, and some others are established in [4, 14]. Together with

the relations established here, the relations between the connectivity measures in the above table are as follows (up to an additive constant):



4 Lower Bounds on Regular Expression Size

Now we have developed enough tools to derive lower bounds on alphabetic width in terms of star height.

Theorem 20. *Let L be a regular language. Then* $\mathrm{alph}(L) \geq 2^{\frac{1}{3}h(L)} - 1$.

Proof. Let r be a regular expression of alphabetic width $n = \mathrm{alph}(L)$. Then the construction given in [21] shows how to transform this expression into an equivalent nondeterministic finite automaton A with ε-transitions having at most $n + 1$ states. It can be readily seen that the digraph underlying the transition structure of the constructed automaton has undirected treewidth at most 2, its undirected version being a series-parallel graph. By Corollary 14, $cr(A) \leq 3\log(n+1)$. The proof is completed by using Theorem 2. □

Again this bound is tight up to a constant factor: Define the language L_n inductively by $L_1 = \varepsilon$, and $L_i = (0L_{i-1}1)^*$ for $i > 1$. Then $\mathrm{alph}(L_n)$ is clearly at most $2n$, but it is known that that $L_{2^k} = k$ for each $k \geq 1$ [28]. Another example is found in [10] with a family of languages K_n having star height n and alphabetic width at most 4^n. Clearly, there cannot exist an upper bound on the alphabetic width in terms of star height, since all finite languages have star height 0, but there are only finitely many languages of bounded alphabetic width.

4.1 Lower Bounds on Alphabetic Width of Language Operations

As a first application of Theorem 20, we exhibit a family of languages over a binary alphabet that shows that several natural operations on regular languages such as complement, intersection and shuffle cannot be supported efficiently by regular expressions; most notably, complementation can require an almost doubly-exponential blow-up in regular expression size. These languages have an appealingly simple structure, and their star height was already studied, although not completely determined, in the very first paper on star height of regular languages [9].

Theorem 21. *For $m, n \in \mathbb{N}$, define $K_m = \{\, w \in \{a,b\}^* \mid |w|_a \equiv 0 \mod n \,\}$ and $L_n = \{\, w \in \{a,b\}^* \mid |w|_b \equiv 0 \mod m \,\}$. Then*

$$h(K_m \cap L_n) = \begin{cases} m & \text{for } m = n \\ \min(m,n) + 1 & \text{otherwise} \end{cases}$$

Proof. The stated upper bound on the star height is proved already in [9, Corollary 2, pp. 394f.], so it remains to show a matching lower bound. It is straightforward to construct deterministic finite automata with m (n, respectively) states describing the languages K_m and L_n,

respectively. By applying the standard product construction on these automata, we obtain a deterministic finite automaton A accepting the language $K_m \cap L_n$. It is not hard to see that this automaton is a minimal trim deterministic finite automaton, and furthermore that it is bideterministic. Therefore Theorem 4 is applicable, and $h(K_m \cap L_n) = cr(A)$.

The directed graph underlying the transition structure of A is isomorphic to the directed discrete $(m \times n)$-torus $T_{m,n}$. So it remains to give a lower bound on the cycle rank of this digraph. By Theorem 17, the cycle rank of each digraph is bounded below by its entanglement, and the entanglement of the directed torus is determined in [5] as m if $m = n$, and $\min(m, n)+1$ otherwise. □

Together with Theorem 20, we immediately obtain some results about the alphabetic width of operations on regular languages. The classical way to extend the syntax of regular expressions is to allow intersection, thus obtaining the semi-extended regular expressions, or to allow complement, resulting in extended regular expressions. Note that the latter can easily simulate the former since intersection can be simulated by the complement and union operators, such that the increase in size of description is within a constant factor.

It is known that semi-extended regular expressions can be exponentially more succinct even than nondeterministic finite automata, and hence than regular expressions. The former fact no longer holds if the number of occurrences of the intersection operator is bounded. But for regular expressions, a *single* intersection operation can infer a huge blow-up in the needed description size:

Theorem 22. *For every $m \geq n$, there exist languages L_m and L_n over a binary alphabet such that, while* $\mathrm{alph}(L_m) \leq m$ *and* $\mathrm{alph}(L_n) \leq n$*, for the intersection of L_m and L_n holds*

$$\mathrm{alph}(L_m \cap L_n) \geq 2^{\frac{1}{6}(n-1)} - 1.$$

Proof. Let $k = \lfloor \frac{m}{2} \rfloor$ and $\ell = \lfloor \frac{n}{2} \rfloor$ Our witness languages are the languages K_k and L_ℓ in the statement of Theorem 21. These languages obviously have regular expressions of alphabetic width at most m and n, respectively. In contrast, the intersection of these languages has star height at least ℓ by Theorem 21. By Theorem 20, we obtain $\mathrm{alph}(L_m \cap L_n) \geq 2^{\frac{1}{3}\ell} - 1 \geq 2^{\frac{1}{6}(n-1)} - 1$. □

Another natural language operation is the shuffle of two languages, which arises in modeling the interleaving the action traces of two processes. The shuffle of two languages L_1 and L_2 over alphabet Σ is

$$\{\, w \in \Sigma^* \mid w \in x \sqcup\!\sqcup y \text{ for some } x \in L_1 \text{ and } y \in L_2 \,\},$$

where the shuffle of two words x and y is defined as

$$\{\, x_1 y_1 x_2 y_2 \dots x_n y_n \mid x = x_1 \dots x_n, y = y_1 \dots y_n, x_i, y_i \in V^*, 1 \leq i \leq n, n \geq 1 \,\}$$

and is denoted by $x \sqcup\!\sqcup y$.

While the shuffle operation preserves regularity, it is known that regular expressions extended with the shuffle operator can be exponentially more succinct than regular expressions—in fact, the same holds for nondeterministic finite automata: As noted in [26], the set of permutations of the string $a_1 a_2 \cdots a_n$ can be described as $a_1 \sqcup\!\sqcup a_2 \sqcup\!\sqcup \cdots \sqcup\!\sqcup a_n$, while every

nondeterministic finite automaton accepting this set and hence every equivalent regular expression has size exponential in n. As detailed in the mentioned work, a suitable encoding of this language gives an exponential lower bound even for binary alphabets. Here we show that a similar blow-up can be caused even by a *single application* of the shuffle operator:

Corollary 23. *For every $m \geq n$, there exist languages L_m and L_n over a binary alphabet such that, while* $\mathrm{alph}(L_m) \leq m$ *and* $\mathrm{alph}(L_n) \leq n$, *for the shuffle of L_m and L_n holds*

$$\mathrm{alph}(L_m \sqcup\!\sqcup L_n) \geq 2^{\frac{1}{3}n} - 1.$$

Proof. Let $K_m = (a^m)^*$ and $L_n = (b^n)^*$. Then $\mathrm{alph}(K_m) = m$ and $\mathrm{alph}(L_n) = n$, but note that $K_m \sqcup\!\sqcup L_n$ equals the language

$$\{\, w \in \{a,b\}^* \mid |w|_a \equiv 0 \mod n \,\} \cap \{\, w \in \{a,b\}^* \mid |w|_b \equiv 0 \mod m \,\},$$

which was already used in the previous lower bounds argument. □

For the complementation operation, we even obtain a lower bound that is roughly doubly exponential:

Theorem 24. *There exists an infinite family of languages L_r over a binary alphabet $\{a,b\}$ such that* $\mathrm{alph}(L_r) \leq r$, *but for the complement holds*

$$\mathrm{alph}(\{a,b\}^* \setminus L_r) = 2^{2^{\Omega(\sqrt{r \log r})}}.$$

Proof. We will show that for infinitely many numbers τ, the complement of $K_\tau \cap L_\tau$ can be described by a small regular expression —recall that Theorem 22 gives a lower bound on the alphabetic width of this language. For $n \in \mathbb{N}$, let $\sigma = \sigma_n = \sum_{i=0}^{n} p_i$ and $\tau = \tau_n = \prod_{i=1}^{n} p_i$ denote the sum and product of the first n primes, respectively. For the proof we use similar ideas as used in the proof of [12, Theorem 27], where it was shown that there are unary regular languages of alphabetic width $O(\sigma_n)$, whose complement has alphabetic width $\Omega(\tau_n)$.

By the Chinese Remainder Theorem (see, e.g., [2]), the set

$$K_\tau = \{\, w \in \{a,b\}^* \mid |w|_a \equiv 0 \mod \tau \,\}$$

equals $\bigcap_{i=1}^{n} K_{p_i}$, and similar, the set

$$L_\tau = \{\, w \in \{a,b\}^* \mid |w|_b \equiv 0 \mod \tau \,\}$$

equals $\bigcap_{i=1}^{n} L_{p_i}$. By duality, the complement $\neg\,(K_\tau \cap L_\tau)$ can be written as

$$\neg\,(K_\tau \cap L_\tau) = \bigcup_{i=1}^{n} \neg K_{p_i} \cup \bigcup_{i=1}^{n} \neg L_{p_i} \tag{1}$$

Next, we give an upper bound on the size of a minimum regular expression for $\neg\,(K_\tau \cap L_\tau)$. For every $m \geq 2$, the language $\neg L_m$ is obtained from $\neg K_m$ by exchanging the roles of the symbols a and b, hence they are of equal alphabetic width, and it is sufficient to give an upper bound for $\mathrm{alph}(\neg K_m)$. The set $\neg K_m$ can be written as $R_m(K_m)^*$, where R_m is the language described

by the regular expression $b^*a(b^*(a+\varepsilon))^{m-1}$, and K_m by the expression $((b^*a)^m)^*$, and thus for every $m \geq 2$ holds $\text{alph}(\neg K_m) \leq 4m$. By Equation 1, we obtain

$$\text{alph}\left(\neg\left(K_\tau \cap L_\tau\right)\right) \leq 2\sum_{i=0}^{n} 4p_i = 8\sigma_n. \tag{2}$$

Using a result given in [2], $\sigma_n \sim \frac{1}{2}n^2 \ln n$, thus for the size r of the expression given above holds $r = 8\sigma_n = O(n^2 \log n)$. By the Prime Number Theorem,

$$\tau_n = e^{(1+o(1))n} = 2^{\Omega(\sqrt{r \log r})}.$$

Finally, Theorem 22 shows that, $K_\tau \cap L_\tau$ has alphabetic width at least $2^{\frac{1}{3}\tau_n} = 2^{2^{\Omega(\sqrt{r \log r})}}$. □

4.2 A Lower Bound for Converting DFAs into Regular Expressions.

From the results in the previous chapter, it can be deduced that there are very simple examples of languages over a binary alphabet for which a blow-up in size of $2^{\Omega(\sqrt{n})}$ is inevitable when converting from an n-state DFA to an equivalent regular expression. The obvious question is now if a lower bound of $2^{\Omega(n)}$ can be reached over a constant alphabet. This section is devoted to giving an affirmative answer.

By Corollary 16, the cycle rank of a loop-free undirected graph G (understood as a symmetric directed graph) can be described in terms of the immutable cops and strong visible robber game. Note that in this case every connected component is also strongly connected. The *greedy* strategy for the robber player is to choose in each step the largest connected component he can reach in the remaining graph. We will identify a class of graphs in which the greedy strategy is particularly successful, namely expander graphs.

Definition 25. *Let $G = (V, E)$ be an undirected graph. For a subset $U \subset V$, the (vertex) boundary of U, denoted by δU, is defined as*

$$\delta U = \{\, v \in V - U \mid \{u, v\} \in E \text{ for some } u \in U \,\}.$$

An (undirected) d-regular graph $G = (V, E)$ with n vertices is called a (n, d, c)-expander (for $c > 0$), if each subset $U \subset V$ of vertices satisfies

$$|\delta U| > c \cdot |U|, \text{ if } |U| < n/2 \quad \text{and} \quad |\delta U| \geq c \cdot (n - |U|), \text{ if } |U| \geq n/2. \tag{3}$$

A now standard probabilistic argument, originally from [32], shows that expander graphs are the rule rather than the exception among d-regular graphs, for all $d \geq 3$. Explicit constructions are also known, see, e.g., [1].

Theorem 26 ([32])**.** *There exists a fixed $c > 0$ such that for any $d \geq 3$ and even integer n, there is an (n, d, c)-expander, which is furthermore d-edge-colorable* [2].

The proof of the following theorem is similar to the proof of [6, Theorem 4], where it was shown that each directed expander graph contains a long directed path.

[2] That is, one an assign to its edges d colors such that no pair of incident edges receives the same color.

Theorem 27. *For $n \geq 3$, let G be a (n, d, c)-expander. Then*

$$cr(G) \geq \frac{c}{d+1}(n-1).$$

Proof. We show that at least $\frac{c}{d+1} \cdot (n-1) - 1$ cops are needed in the immutable cops and strong visible robber game on G in order to catch the robber. It is no restriction to require that the set X_0 of initial positions for the cop player is empty, and that the cop player places exactly one new cop in the first and every following round. That is, if X_i denotes the set of positions occupied by the cops in round i, we have $|X_0| = 0$ and $|X_{i+1}| = |X_i| + 1$ for $i \geq 0$. It is convenient to think of the robber's moves as choosing a (strongly) connected component C_i in round i, instead of choosing a position $v_i \in C_i$. G being a connected undirected graph because of the expansion property, we have $C_0 = V$. It is also easy to see that $C_{i+1} \subseteq C_i$.

For $i \geq 0$, consider the $(i+1)$-th move of the robber. The strategy we use for the robber is a greedy choice, based on the size of the maximum strongly connected component in the remaining graph. A more precise description follows. If the cop player places no cop on component C_i, the robber player stays where he is. So assume x_{i+1}, the position for the $(i+1)$-th cop, is in C_i. By removing x_{i+1} from the graph $G[C_i]$, the remaining graph falls apart into connected components $C_{i+1}^{(1)}, C_{i+1}^{(2)}, \ldots C_{i+1}^{(d')}$. Since G is d-regular, we have $d' \leq d$. Among these, the robber player chooses the component of maximum size. Since $\bigcup_{j=1}^{d'} G[C_{i+1}^{(j)}] = G[C_i] - x_{i+1}$, and the largest connected component C_{i+1} has at least average size, we have

$$|C_{i+1}| \geq \frac{|C_i| - 1}{d}. \tag{4}$$

The next question is to locate the vertex boundary of C_{i+1}. It is clear that $\delta C_{i+1} \cap C_{i+1}^{(j)} = \emptyset$ holds for all j, since the graphs $G[C_{i+1}^{(j)}]$ are mutually disconnected in $G[C_i] - x_{i+1}$. Neither can there be edges between C_{i+1} and any connected component $C_k^{(j)}$ split off in any previous round $k \leq j$, since C_{i+1} is a subset of C_k, the component chosen in round k. So, where is the vertex boundary of δC_{i+1}? The only candidates remaining are the vertices X_{i+1} occupied by the cops in round $i+1$, so $\delta C_{i+1} \subseteq i+1$. We note for later reference the inequality

$$|\delta C_{i+1}| \leq i + 1. \tag{5}$$

We now divide the game into two phases. The first phase lasts as long as $|C_{i+1}| \geq \frac{n}{2}$, or, to put it differently, the second phase starts in round i, if $|C_i| \geq \frac{n}{2}$ and $|C_{i+1}| < \frac{n}{2}$. We will prove that this first phase lasts for quite a few steps. Here the expander inequalities come into play: During the first phase, that is, as long as $|C_{i+1}| \geq \frac{n}{2}$, we have

$$|\delta C_{i+1}| \geq c \cdot (n - |C_{i+1}|) \tag{6}$$

With Ineq. (5), we obtain thus

$$|C_{i+1}| \geq n - \frac{i+1}{c}, \tag{7}$$

during the first phase. We now claim that the first phase lasts at least as long as the following inequality holds:

$$i + 1 \leq c \cdot \frac{|C_i| - 1}{d}. \tag{8}$$

Namely, we have by Ineq. (5) and Ineq. (4) that

$$|\delta C_{i+1}| \leq i+1 \leq c \cdot \frac{|C_i|-1}{d} \leq c|C_{i+1}|, \tag{9}$$

contradicting the expander inequality which has to hold in the second phase.

To give a lower bound on the starting time of the second phase, we determine the largest m such that Ineq. (8) still holds, thus guaranteeing that the first phase is not yet over after at most m steps. Determining m over the real domain is more convenient, so let $C(x)$ be a continuous, monotonically decreasing, nonnegative function with $C(i) = |C_i|$ for integer values i.

Ineq. (8) becomes equality for $m+1 = \frac{c}{d}(C(m)-1)$, and resolving after $C(m)$ gives $C(m) = \frac{d(m+1)}{c}+1$. By Ineq. (7), we have $C(m) \geq n - \frac{m}{c}$. Plugging the value of $C(m)$ into this inequality and resolving for m, we get

$$m \geq \frac{c(n-1)}{d+1} - \frac{d}{d+1} > \frac{c}{d+1}(n-1) - 1.$$

Recall that $C(m) \geq \frac{n}{2}$, thus $C(m) \geq 2$ for $n \geq 4$. It is easy to see that at least two cops are needed to catch the robber on a connected undirected graph of size at least two, thus a winning strategy for the cops on G is ruled out unless at least $\frac{c}{d+1}(n-1)+1$ cops are used. By Corollary 16, we see that the cycle rank of G is at least $\frac{c}{d+1}(n-1)$. □

Lemma 28. *For every d-edge colorable, connected undirected graph G with n vertices of cycle rank k, there exists an n-state deterministic finite automaton A over a d-symbol alphabet such that the star height of $L(A)$ is k.*

Proof. Let $G = (V, E)$ be such a graph, with $V = \{1, 2, \ldots, n\}$. and maximum degree d, equipped with an edge coloring $c : E \to \{0, \ldots, d\}$ such that no pair of incident edges receive the same color. Given this colored graph, we construct a DFA over the alphabet $\Sigma = \{a_1, \ldots, a_d\}$ with state set V, start and single final state $v_0 \in V$ (arbitrary), and whose transition relation is defined as follows: $\delta(p, a_i) = q$ if the colored graph G has an i-colored edge $\{p, q\}$. It is not hard to see that this automaton is a trim bideterministic automaton, and therefore minimal. Furthermore, its underlying directed graph is symmetric, and its undirected version is isomorphic to G. Thus by Theorem 4, the star height of $L(A)$ equals k. □

This last lemma, together with Theorems 20, 26, and 27 gives us the first main result of this section:

Theorem 29. *For alphabet size $|\Sigma| \geq 3$, there is an infinite family of languages L_n such that $sc(L_n) \leq n$, but $\mathrm{alph}(L_n) = 2^{\Omega(n)}$.* □

This already gives an affirmative answer to Open Problem 3 in [12], which asked whether such a family of languages exists, over some constant alphabet. We can even give examples over a binary alphabet, by a suitable binary encoding. Some care has to be taken, however, since we want to choose a homomorphism that preserves star height. The existence of reasonably economic binary encodings with this property have been already conjectured in [9], their existence was proved in [28], and a precise characterization of homomorphisms preserving star height was given in [17]. Here we only need the following fact from [28]:

Theorem 30. *Let $\Sigma = \{a_1, a_2, \ldots, a_d\}$ be a finite alphabet, $d \geq 1$, and let $\Delta : \Sigma \to \{a, b\}$ be the homomorphism defined by $\Delta(a_i) = a^i b^{d-i+1}$, for $i = 1, 2, \ldots, d$. Then for every regular language $L \subseteq \Sigma^*$ the star height of L equals the star height of $\Delta(L)$.*

This allows us for coding down to a binary alphabet, and we obtain:

Theorem 31. *For alphabet size $|\Sigma| \geq 2$, there is an infinite family of languages K_n such that $sc(K_n) \leq n$, but $\mathrm{alph}(K_n) = 2^{\Omega(n)}$.*

Proof. We encode the language family L_n from Theorem 29 in binary using the star height preserving homomorphism Δ presented above. An n-state deterministic finite automaton A accepting L_n can be easily transformed into a $(4+\sum_{i=1}^{3} i)n$-state deterministic finite automaton A' by replacing each state q with an appropriate set of "private" states for q such that whenever A moves from q to state r on input a_i, automaton A' moves from q to state r on input $\Delta(a_i)$ using the private state set of q. The number of states in A' is still in $O(n)$, and the star height is preserved by the homomorphism, hence with Theorem 20 we obtain $\mathrm{alph}(\Delta(L_n)) = 2^{\Omega(n)}$. □

References

[1] Noga Alon, Oded Schwartz, and Asaf Shapira. An elementary construction of constant-degree expanders. In Nikhil Bansal, Kirk Pruhs, and Clifford Stein, editors, *ACM-SIAM Symposium on Discrete Algorithms 2007*, pages 454–458. SIAM, 2007.

[2] E. Bach and J. Shallit. *Algorithmic number theory, volume 1: Efficient algorithms*. MIT Press, 1996.

[3] J. Barát. Directed path-width and monotonicity in digraph searching. *Graphs and Combinatorics*, 22(2):161–172, 2006.

[4] D. Berwanger, A. Dawar, P. Hunter, and S. Kreutzer. Dag-width and parity games. In B. Durand and W. Thomas, editors, *Symposium on Theoretical Aspects of Computer Science 2006*, volume 3884 of *LNCS*, pages 524–536. Springer, 2006.

[5] D. Berwanger and E. Grädel. Entanglement – A measure for the complexity of directed graphs with applications to logic and games. In F. Baader and A. Voronkov, editors, *Logic for Programming, Artificial Intelligence, and Reasoning 2004*, volume 3452 of *LNCS*, pages 209–223. Springer, 2005.

[6] A. Björklund, T. Husfeldt, and S. Khanna. Approximating longest directed paths and cycles. In J. Díaz, J. Karhumäki, A. Lepistö, and D. Sannella, editors, *International Colloquium on Automata, Languages and Programming*, volume 3142 of *LNCS*, pages 222–233. Springer, 2004.

[7] H. L. Bodlaender, J. R. Gilbert, H. Hafsteinsson, and T. Kloks. Approximating treewidth, pathwidth, frontsize, and shortest elimination tree. *Journal of Algorithms*, 18(2):238–255, 1995.

[8] R. S. Cohen. Star height of certain families of regular events. *Journal of Computer and System Sciences*, 4(3):281–297, 1970.

[9] L. C. Eggan. Transition graphs and the star height of regular events. *Michigan Mathematical Journal*, 10:385–397, 1963.

[10] A. Ehrenfeucht and H. P. Zeiger. Complexity measures for regular expressions. *Journal of Computer and System Sciences*, 12(2):134–146, 1976.

[11] S. C. Eisenstat and J. W. H. Liu. The theory of elimination trees for sparse unsymmetric matrices. *SIAM Journal on Matrix Analysis and Applications*, 26(3):686–705, 2005.

[12] 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.

[13] W. Evans, P. Hunter, and M. A. Safari. D-Width and cops and robbers. Submitted, 2007.

[14] H. Gruber. On the D-width of directed graphs. Manuscript, December 2007.

[15] H. Gruber and J. Johannsen. Optimal lower bounds on regular expression size using communication complexity. In R. Amadio, editor, *Foundations of Software Science and Computation Structures*, volume 4962 of *LNCS*, pages 273–286. Springer, 2008. to appear.

[16] K. Hashiguchi. Algorithms for determining relative star height and star height. *Information and Computation*, 78(2):124–169, 1988.

[17] K. Hashiguchi and N. Honda. Homomorphisms that preserve star height. *Information and Control*, 30(3):247–266, 1976.

[18] J. E. Hopcroft and J. D. Ullman. *Introduction to Automata Theory, Languages and Computation.* Addison-Wesley, 1979.

[19] P. Hunter. Losing the +1: Directed path-width games are monotone. Manuscript, 2006.

[20] P. Hunter and S. Kreutzer. Digraph measures: Kelly decompositions, games and orderings. preliminary version appeared at SODA '07. To appear, 2007.

[21] Lucian Ilie and Sheng Yu. Follow automata. *Information and Computation*, 186(1):140–162, 2003.

[22] T. Johnson, N. Robertson, P. D. Seymour, and R. Thomas. Directed tree-width. *Journal of Combinatorial Theory, Series B*, 82(1):138–154, 2001.

[23] D. Kirsten. Distance desert automata and the star height problem. *RAIRO – Theoretical Informatics and Applications*, 39(3):455–509, 2005.

[24] S. C. Kleene. Representation of events in nerve nets and finite automata. In C. E. Shannon and J. McCarthy, editors, *Automata Studies*, Annals of Mathematics Studies, pages 3–42. Princeton University Press, 1956.

[25] A. Martinez. Efficient computation of regular expressions from unary NFAs. In J. Dassow, M. Hoeberechts, H. Jürgensen, and D. Wotschke, editors, *Workshop on Descriptional Complexity of Formal Systems 2002*, pages 216–230, London, Canada, 2002.

[26] A. J. Mayer and L. J. Stockmeyer. Word problems – This time with interleaving. *Information and Computation*, 115(2):293–311, 1994.

[27] R. McNaughton. The loop complexity of pure-group events. *Information and Control*, 11(1/2):167–176, 1967.

[28] R. McNaughton. The loop complexity of regular events. *Information Sciences*, 1:305–328, 1969.

[29] P. H. Morris, R. A. Gray, and R. E. Filman. Goto removal based on regular expressions. *Journal of Software Maintenance*, 9(1):47–66, 1997.

[30] J. Nešetřil and P. Ossona de Mendez. Tree-depth, subgraph coloring and homomorphism bounds. *European Journal of Combinatorics*, 27(6):1022–1041, 2006.

[31] J. Obdržálek. Dag-width: Connectivity measure for directed graphs. In *ACM-SIAM Symposium on Discrete Algorithms*, pages 814–821. ACM Press, 2006.

[32] Mark S. Pinsker. On the complexity of a concentrator. In *Annual Teletraffic Conference*, pages 318/1–318/4, 1973.

[33] N. Robertson and P. D. Seymour. Graph minors. II. Algorithmic aspects of tree-width. *Journal of Algorithms*, 7(3):309–322, 1986.

[34] M. A. Safari. D-Width: A more natural measure for directed tree width. In J. Jedrzejowicz and A. Szepietowski, editors, *Mathematical Foundations of Computer Science 2005*, volume 3618 of *LNCS*, pages 745–756. Springer, 2005.

[35] J. Sakarovitch. The language, the expression, and the (small) automaton. In Jacques Farré, Igor Litovsky, and Sylvain Schmitz, editors, *Conference on Implementation and Application of Automata*, volume 3845 of *Lecture Notes in Computer Science*, pages 15–30, 2005.

[36] G. Schnitger. Regular expressions and NFAs without ε-transitions. In B. Durand and W. Thomas, editors, *Symposium on Theoretical Aspects of Computer Science*, volume 3884 of *Lecture Notes in Computer Science*, pages 432–443, 2006.

[37] R. Schreiber. A new implementation of sparse Gaussian elimination. *ACM Transactions on Mathematical Software*, 8(3):256–276, 1982.

[38] H. Tamm and E. Ukkonen. Bideterministic automata and minimal representations of regular languages. *Theoretical Computer Science*, 328(1-2):135–149, 2004.

[39] V. Waizenegger. Über die Effizienz der Darstellung durch reguläre Ausdrücke und endliche Automaten. Diplomarbeit, Rheinisch-Westfälische Technische Hochschule Aachen, Fachbereich Informatik, 2000.