Verifying Proofs in Constant Depth*

Olaf Beyersdorff[1] Samir Datta[2] Andreas Krebs[3]
Meena Mahajan[4] Gido Scharfenberger-Fabian[5]
Karteek Sreenivasaiah[4] Michael Thomas[6]
Heribert Vollmer[7]

[1]School of Computing, University of Leeds, UK
[2]Chennai Mathematical Institute, India
[3]University of Tübingen, Germany
[4]Institute of Mathematical Sciences, Chennai, India
[5]Mathematical Institute, University of Potsdam, Germany
[6]TWT GmbH, Neuhausen a. d. F., Germany
[7]Institute for Theoretical Computer Science, Leibniz University Hanover, Germany

January 3, 2013

Abstract

In this paper we initiate the study of proof systems where verification of proofs proceeds by NC^0 circuits. We investigate the question which languages admit proof systems in this very restricted model. Formulated alternatively, we ask which languages can be enumerated by NC^0 functions. Our results show that the answer to this problem is not determined by the complexity of the language. On the one hand, we construct NC^0 proof systems for a variety of languages ranging from regular to NP complete. On the other hand, we show by combinatorial methods that even easy regular languages such as Exact-OR do not admit NC^0 proof systems. We also show that Majority does not admit NC^0 proof systems. Finally, we present a general construction of NC^0 proof systems for regular languages with strongly connected NFA's.

1 Introduction

The notion of a proof system for a language L was introduced by Cook and Reckhow in their seminal paper [13] as a function f that has as its range exactly all strings of L. Pre-images of f are considered as proofs for elements

*Research supported by a DAAD/DST grant, DFG grant VO 630/6-2, and by a grant from the John Templeton Foundation. A preliminary version containing some of the results from this paper appeared in the proceedings of MFCS'11 [8].

$x \in L$. In addition, one needs to guarantee that proofs can be verified efficiently. In the model of Cook and Reckhow [13] this is captured by requiring that f is polynomial-time computable, leading to the notion of polynomial-time computable proof systems. In this setting, finding a proof might be difficult, but verifying the validity of a proof can be done efficiently. In the last decades, polynomial-time computatble proof systems were deeply studied in the field of proof complexity and a rich body of results is known regarding the complexity of proofs for concrete proof systems (cf. [22] for a survey).

Recently, there has been great interest in understanding the power of proof systems that use stronger computational resources to verify proofs. In this direction, Pudlák [21] studies quantum proof systems, Cook and Krajíček [12] introduce proof systems that may use a limited amount of non-uniformity (see also [9, 10]), and Hirsch and Itsykson [18, 19] consider proof systems that verify proofs with the help of randomness. In this research, the original Cook-Reckhow framework is generalized and exciting results are obtained about the strength and the limitations of theorem proving with respect to these powerful models.

In this work we take the opposite approach and ask for minimal resources that suffice to verify proofs. Our starting point is the observation that every polynomial-time computable proof system in the Cook-Reckhow model is efficiently simulated (*i.e.*, p-simulated[1]) by a proof system where verification of proofs proceeds in AC^0. This immediately leads to the question whether even less powerful computational resources are sufficient. Our investigation focuses on NC^0 circuits—Boolean circuits of constant depth over NOT gates and bounded fan-in AND and OR gates—which constitute one of the weakest computational models in computational complexity. In a related approach, Goldwasser et al. [16] recently studied proof verification by NC^0 circuits in the context of interactive proof systems.

The restrictions imposed by the NC^0 model are so severe that a similar result as the mentioned one for AC^0 fails drastically. NC^0-computable proof systems are functions which shrink the input by at most a constant factor. Thus every language with an NC^0 proof system is computable in nonuniform nondeterministic linear time. We therefore concentrate on the question which languages admit NC^0 proof systems, *i.e.*, which languages can be *enumerated* by families of NC^0 circuits.

A related line of research studies NC^0-computable functions in a cryptographic context [5,6,14,17,20]. One of the main problems in this area is to construct pseudorandom generators which are computed by NC^0 circuits [5,6,14,20]. This question asks for NC^0-computable functions for which the range is hard to distinguish from a uniform distribution. In another thread [23,24], for a function f, one is interested in the uniform distribution over $\langle x, f(x)\rangle$. Sampling exactly from this distribution may be possible even if computing $f(x)$ is hard. However, it is shown that for some functions (such as detecting exact-Hamming-weight αn), getting even close to the uniform distribution via specific types of NC^0

[1] If f and g are proof systems for L, then f *p-simulates* g if every g-proof w can be transformed in polynomial time into an f-proof w' with $g(w) = f(w')$ [13].

circuits (d-local circuits) is not possible. In contrast, we are looking here at the related, but possibly easier problem to understand which sets can appear at all as the range of NC^0-computable functions, irrespective of the distribution on the support. While our lower bounds indicate that sampling the distribution exactly is not possible for some functions, it does not say anything about how close to the uniform distribution we can get.

We note that Cryan and Miltersen [14] exhibit an NC^0-computable function whose range is NP complete. Thus, there are NP-complete languages that admit an NC^0 proof system. Our results, however, indicate that the answer to the question of the existence of such a proof system does not correlate with the computational complexity of the target language. In our first contribution, we construct NC^0 proof systems for a variety of natural problems, including regular, NC^1-complete, and P-complete languages. In addition, we exhibit a general construction for NC^0 proof systems which works for all regular languages that are accepted by a strongly connected NFA. Our construction directly transforms this NFA into an NC^0 proof system.

Secondly, we demonstrate that there even exist regular languages which do not admit NC^0 proof systems. We also show that the canonical threshold language MAJ does not admit NC^0 proof systems. The proof techniques we use are combinatorial arguments tailored towards our specific problems.

Taken together, both lines of the results presented here show how different the study of NC^0-enumerability is from complexity considerations of decision problems. In particular, it is not clear if lower bound techniques which are used against restricted circuit classes (cf. [25, 26]) are applicable to show lower bounds for NC^0 proof systems.

This paper is organized as follows. We start in Section 2 by defining the concept of NC^0 proof systems and make some initial observations. In Section 3 we construct NC^0 proof systems for several languages of different type. This is followed by Section 4 where we develop a lower bound technique for the depths of NC circuit enumerations of several easy languages including Exact-OR and some threshold functions. In Section 5 we show by an independent technique that the language MAJ does not admit NC^0 proof systems. In Section 6 we generalize some of the ideas for NC^0 proof systems from Section 3 to obtain proof systems for large classes of regular languages. Finally, we conclude in Section 7 with some discussion and future perspectives.

2 Definitions

A family of Boolean circuits (see, e.g., [25]) $\left(C_n\right)_{n\geq 1}$ is said to be a proof system for a function $f\colon \{0,1\}^* \longrightarrow \{0,1\}$ if it satisfies the following conditions:

1. Output length: For some function $m : \mathbb{N} \longrightarrow \mathbb{N}$, and for all $n \geq 1$,
   - If $f^{-1}(1) \cap \{0,1\}^n \neq \emptyset$, then $C_n\colon \{0,1\}^{m(n)} \to \{0,1\}^n$.
   - Otherwise C_n is empty.

2. Soundness: For all n where C_n is non-empty, and for all words $x \in \{0,1\}^{m(n)}$, $C_n(x) \in f^{-1}(1)$. (That is, $f(C_n(x)) = 1$.)

3. Completeness: For all $y \in f^{-1}(1) \cap \{0,1\}^n$, there is a word $x \in \{0,1\}^{m(n)}$ such that $C_n(x) = y$; we say that x is a *proof* of the word y in the pre-image of 1 under f.

4. For some functions $s, d : \mathbb{N} \longrightarrow \mathbb{N}$, each C_n has size $s(n)$, depth $d(n)$, and is built using AND, OR, and NOT gates.

That is, the circuit family has as its range exactly the set $f^{-1}(1)$.

The circuit family is said to be a constant-depth proof system if the AND and OR gates have unbounded fan-in, and for some constant d and for all n, $d(n) \leq d$. A constant-depth proof system of polynomial size (for some constant c and for all n, $s(n) \leq n^c$) is said to be an AC^0 proof system. Note that the size bound is non-standard: it is measured in terms of the output length, not input length.

The circuit family is said to be an NC^0 proof system if the AND and OR gates have bounded fan-in, and for some constant d and for all n, $d(n) \leq d$. This, along with the fan-in bound, implies $s(n) \leq cn$ for some constant c.

If a function $f\colon \{0,1\}^* \longrightarrow \{0,1\}$ has a proof system of a particular type (constant-depth, AC^0, NC^0), then we say that f admits a proof system of that type.

If the circuit family is *uniform*, then we say that the proof system is *uniform*. Here, a uniform circuit family is a family whose direct connection language, i.e., a language describing the structure (nodes, wires, gates types) of the circuits in the family, is decidable. If the direct connection language is decidable in polynomial time, then the family is said to be P-*uniform*. If the language is decidable in logarithmic time, then the family is said to be $\mathsf{DLOGTIME}$-*uniform*. (For more formal definitions, we refer the reader to [25].)

We remark that all lower bounds we will present in the sequel of this paper hold even for nonuniform proof systems, while all upper bounds will yield $\mathsf{DLOGTIME}$-uniform proof systems, unless explicitly stated otherwise.

For a language $L \subseteq \{0,1\}^*$, we say that L admits an NC^0 proof system, or that L is enumerable in NC^0, if its characteristic function χ_L admits such a proof system. In other words, there is an NC^0 circuit family which produces as output all the strings in L and no other strings. As before, if $C(x) = y$, then we view x as a *proof* that $y \in L$.

We observe that uniform AC^0 proof systems do exist for every NP set. In fact, a more general statement is true. (Here uniformity only refers to computable direct connection languages.)

Proposition 2.1 (Folklore)

1. *Every language in* NP *admits a uniform* AC^0 *proof system.*

2. *More generally, every computable language admits a uniform constant depth proof system.*

3. Even more generally, every language admits a constant-depth proof system.

Proof. (Sketch.) For an arbitrary language L with characteristic function $\chi_L = f$, let n be a length where L is non-empty. Pick an arbitrary $w \in L^{=n}$. Restricted to length n, f is computed by a constant-depth Boolean circuit (possibly of size exponential in n) D. The circuit C_n extends D: If on input x, $D(x) = 1$, then output x otherwise output w. Clearly C_n is also constant-depth, and its range is exactly $L^{=n}$, proving (3).

Now let the computable language L be decided by the deterministic Turing Machine M. The run-time of M with respect to the length of the input can be assumed to be computable. Also a default word $w \in L^{=n}$ can be found effectively if it exists. A proof of a word $y \in L$ is an encoding of an accepting sequence of configurations of M on input y. The correctness of such a sequence of configurations can be checked locally, essentially in two consecutive configurations only three letters (around the head position on the tape) can be different. If our circuit reads an input that is not such an encoding, then it outputs some default value $w \in L$ of the appropriate length as above. All of this can be done by a constant depth uniform circuit, proving (2).

Let finally L be accepted in polynomial time by the nondeterministic Turing machine M. Proceeding as above, the checking circuit is of size polynomial in the output word, proving (1). □

As mentioned already in the introduction, Cryan and Miltersen [14] exhibit an NP-complete language that admits even a uniform NC^0 proof system. But it is quite easy to see that this is not the case for every NP language. Indeed, as a consequence of the last condition of the definition above, we see that $m(n) \leq 2^d n \in O(n)$ and the circuits C_n are also of size $O(n)$; each bit of the output depends on $O(1)$ bits of the input proof. Thus if L has NC^0 proof systems, then strings in L have linear-sized proofs that are locally verifiable. This leads to the following observation, which will be considerably strengthened in Section 4.

Proposition 2.2 *There are non-trivial languages in* NP *that do not admit any* DLOGTIME*-uniform* NC^0 *proof system.*

Proof. If a language L has a DLOGTIME-uniform NC^0 proof system, then it can be recognised in NTIME(n): given an input y, guess the linear-sized proof x, evaluate the circuit $C_{|y|}(x)$, and verify that its output is y. But by the nondeterministic time hierarchy we know that NP is not contained in NTIME(n).□

3 Languages with NC^0 proof systems

In this section, we construct NC^0 proof systems for a variety of languages.

We start with an NC^1-complete language that admits an NC^0 proof system. The word problem for a finite monoid M with identity e is (membership in) the language: $\{\langle m_1, m_2, \ldots, m_n\rangle \in M^* : \prod_{i=1}^{n} m_i = e\}$. We assume here that for some constant c depending only on M, each element of M is described by a bit string of exactly c bits.

Proposition 3.1 *The word problem for finite groups admits an* NC^0 *proof system.*

Proof. We describe the circuit $C_n : \{0,1\}^{cn-c} \to \{0,1\}^{cn}$. (Since the word problem contains only words of lengths divisible by c, we produce circuits only for such lengths.) Given the encoding of a sequence $g_1, \ldots, g_{n-1}$, and assuming that $g_0 = g_n = e$, C_n produces the sequence $\langle h_1, \ldots, h_n \rangle$ where $h_i = g_{i-1}^{-1} g_i$. Thus $\prod h_i = e$ and the word is indeed in the language. Conversely, every word $\langle h_1, \ldots, h_n \rangle$ in the language is produced by this circuit on input $g_1, \ldots, g_{n-1}$ where $g_i = \prod_{j \leq i} h_j$. □

Corollary 3.2 *The parity function admits an* NC^0 *proof system.*

In proving Proposition 3.1, we used all the three group axioms: associativity, existence of an identity and existence of inverses. We can relax some of these and still get an NC^0 proof system. For example, the *OR* operation is associative and has an identity, but not all elements do have an inverse. Yet we show that the language $L_{OR} = \{w = w_1 \ldots w_n \in \{0,1\}^* : \bigvee_{i=1}^n w_i = 1\}$ has an NC^0 proof system.

Proposition 3.3 *The language* L_{OR} *admits an* NC^0 *proof system.*

Proof. The circuit $C_n : \{0,1\}^{2n-1} \to \{0,1\}^n$ takes as input bit strings $a = a_1 \ldots a_n$ and $b = b_1 \ldots b_{n-1}$, and outputs a sequence $w = w_1 \ldots w_n$ where

$$(\text{for } 1 \leq i \leq n) \qquad w_i = \begin{cases} a_i & \text{if } (b_{i-1} \vee a_i) = b_i \\ 1 & \text{otherwise.} \end{cases}$$

Here for notational convenience we assume that $b_0 = 0, b_n = 1$. Notice that if each b_i correctly encodes the OR of the prefix $a_1 \ldots a_i$, then $b_n = 1$ ensures that at least one $w_i = a_i$ is 1. Otherwise if there is ever a discrepancy between the b_i's and the prefix OR's of a_j's, we introduce a 1 at w_i; thus w is indeed in L_{OR}. Since for each string $a \in L_{OR}$ there is a correct string b with $b_0 = 0$, $b_i = b_{i-1} \vee a_i$ and $b_n = 1$, every string in L_{OR} is produced by C_n. Thus C_n is an onto map from $\{0,1\}^{2n-1} \to L_{OR} \cap \{0,1\}^n$ completing the proof. □

We next consider another NC^1-complete problem, viz. reachability in bounded width directed acyclic graphs. This example illustrates a proof system, which, for the lack of a better description, we refer to as "input altering proofs".

A layered graph with vertices arranged in layers from $0, 1, \ldots, L$ with exactly W vertices per layer (numbered from $0, \ldots, W-1$) and edges between vertices in layer i to $i+1$ for $i \in \{0, \ldots, L-1\}$ is a positive instance of reachability if and only if there is a directed path from vertex 0 at layer 0 to vertex 0 at layer L. A description of the graph consists of a layer by layer encoding of the edges as a bit vector. In other words it consists of a string $x = x^0 x^1 \ldots x^{L-1} \in (\{0,1\}^{W^2})^L$ where the x^i is indexed by $j, k \in \{0, \ldots, W-1\}$ and $x^i[j,k] = 1$ if and only if the j-th vertex on the i-th layer and the k-th vertex on the $(i+1)$-th layer share an edge. The language L_{BWDR} consists of those strings $x \in (\{0,1\}^{W^2})^L$ which describe a positive instance of reachability, for some $W \in O(1)$. Then we have:

Proposition 3.4 *The language L_{BWDR} admits an NC^0 proof system.*

Proof. The proof consists of a string $x \in (\{0,1\}^{W^2})^L$ which describes the graph and a string $v = v^1 \dots v^{L-1} \in (\{0,1\}^V)^{L-1}$ representing a path. Here $V = \lceil \log W \rceil$ is the number of bits required to describe a vertex at a given layer in binary.

Given x, v we first replace each v^i which occurs in v and which represents a number greater than $W-1$ by the bit string consisting of V zeroes. This requires a circuit of depth $O(\log(V)) = O(\log \log W)$. For the ease of notation we refer to the modified v as v also.

Next, for each $i \in \{0, \dots, L-1\}$, we add the edge represented by (v^i, v^{i+1}) to the graph represented by x by setting $x^i[v^i, v^{i+1}] = 1$. This ensures that the graph contains the path represented by v, i.e. it is a positive instance. Clearly to address the appropriate bits of x^i we need a circuit of depth $O(\log V) = O(\log \log W)$. Finally, we output this modified x. It is easy to see that all positive instances will be output by this circuit for some inputs. Since W is a constant, we will obtain an NC^0 proof system. □

The same idea can be used for addition and comparison. Consider the function $f_+ : \{0,1\}^n \times \{0,1\}^n \times \{0,1\}^{n+1} \to \{0,1\}$ such that $f_+(a,b,s) = 1$ if and only if $A + B = S$ where a, b are the n-bit binary representations of the numbers A, B and s is the $(n+1)$-bit binary representation of S. Also consider the function $f_\leq : \{0,1\}^n \times \{0,1\}^n \to \{0,1\}$ where $f_\leq(a,b) = 1 \iff A \leq B$, where again a, b are the n-bit binary representations of numbers A, B.

Proposition 3.5 *The function f_+ admits an NC^0 proof system.*

Proof. The circuit $C_n : \{0,1\}^{3n} \to \{0,1\}^{3n}$ maps three strings $\alpha = \alpha_{n-1} \dots \alpha_0$, $\beta = \beta_{n-1} \dots \beta_0$ and $\gamma = \gamma_n \dots \gamma_1$ (for notational convenience assume that $\gamma_0 = 0$) to strings a, b, s with the intent that γ will serve as the carry sequence in the grade-school addition of the two numbers α, β. Also, if we ever discover a discrepancy between the assumed carry sequence and the two numbers α, β we correct the error by altering α, β appropriately to yield a, b. So this is an "input-altering proof". Formally, for $0 \leq i \leq n-1$, if $\mathrm{Th}^3_2(\alpha_i, \beta_i, \gamma_i) = \gamma_{i+1}$ then $a_i = \alpha_i, b_i = \beta_i$ otherwise, set a_i, b_i arbitrarily under the constraint that $\mathrm{Th}^3_2(a_i, b_i, \gamma_i) = \gamma_{i+1}$. Also set $s_i = a_i \oplus b_i \oplus \gamma_i$. Set $s_n = \gamma_n$. The input-alteration ensures that the output is always in the language, and for each word $\langle a, b, s \rangle$ in the language, the proof that gives the correct carry sequence ensures that the word is produced as output. □

Proposition 3.6 *The function $f_\leq$ admits an NC^0 proof system.*

Proof. The proof consists of four n-bit strings $\alpha, \alpha', \gamma, \beta$, with the intent that γ is the carry sequence for the sum of α, α' which yields β. Again in the proof we ensure that if the carry bits γ_i, γ_{i-1} are compatible with α, α' summing to β, then copy α_i, β_i to a_i, b_i respectively. Otherwise, set $a_i = 0, b_i = 1$ (which ensures that for $j > i$ if $a_j = b_j$ then $a < b$). As before, the input alteration guarantees soundness, and the proof with the correct carry bits guarantees completeness. □

We now consider a P-complete language, Grid Circuit Value. An instance consists of a planar circuit with vertices embedded in a square grid so that the circuit wires lie only along the grid edges and are directed to go only due east or due north. All possible wires are present. The gates can be arbitrary functions of the two inputs and two outputs. All inputs are present on the outer face of the circuit (i.e. on the southern and western boundaries). It is easy to see that this problem is contained in P. To see that it is P hard, we reduce the Circuit Value Problem to it under, say, DLogspace reductions. First make the circuit planar by using the usual cross-over gadget [15] to remove all crossings. Now, embed the circuit in the grid by using a method similar to the one used in [4, 11] to obtain the required embedding. Finally we replace all missing wires by altering the gates to ignore the value from any missing input and output an arbitrary value, say zero along all missing outputs.

Using the strategy of locally correcting the input if the proof shows an inconsistency, we can show the following:

Proposition 3.7 *The Grid Circuit Value Problem admits an* NC^0 *proof system.*

Proof. The proof consists of a string describing the circuit, that is, the truth tables of (both outputs) of a gate for each gate position and a value for each of the wires in the circuit. Since each truth table is for a 2-input and 2-output gate, it is represented by a truth table of 8 bits. Thus for a grid consisting of n vertices on each side, with m input variables, the input string is $(g, v) \in \{0,1\}^{8n^2} \times \{0,1\}^{2(n-1)n}$. The output of the circuit is a pair $(g', x, b) \in \{0,1\}^{8n^2} \times \{0,1\}^{2n-1} \times \{0,1\}$ with g' describing new truth tables obtained by copying those from g already consistent with v, and modifying the others to make them consistent with the values in v (this is always possible by setting one entry of each inconsistent truth-table). The string x describes the values (from v) corresponding to inputs and b the value of the output gate. □

Remark 3.8 As mentioned earlier, Cryan and Miltersen [14] (and in fact already Agrawal et al. [3]) show that a certain NP-complete language admits an NC^0 proof system. The language they consider is just an encoding of 3-SAT: for each n, instances with n variables are encoded by an $M = 2^3\binom{n}{3}$ bit string, where each bit indicates whether the corresponding potential clause is present in the instance. A proof consists of an assignment to the propositional variables and a suggestion for a 3-CNF, which is modified by the proof system in order to be satisfied by the given assignment. The clause bit is flipped if (and only if) (1) it is on, and (2) the clause is not satisfied by the assignment. Since each potential clause has its reserved "indicator bit", checking whether the clause is satisfied by the assignment requires looking at exactly three fixed bits of the assignment. It is easy to see that this system generates exactly the satisfiable 3-SAT instances.

Next, we describe some generic constructions and closures. They are easy to see, but we state them explicitly for later use.

Lemma 3.9 *Let w be any fixed string, and let L be any language. Then L admits an NC^0 system if and only if $L \cdot \{w\}$ does.*

Lemma 3.10 *If $A, B \subseteq \{0,1\}^*$ admit NC^0 proof systems, then so does $A \cup B$.*

Proof. Let the proof systems for A and B be witnessed by circuit families C' and C'', with proof lengths $m'(n)$ and $m''(n)$ respectively. We construct the circuit family C for $A \cup B$, with proof length $m'(n) + m''(n) + 1$, as follows: C_n consists of a copy of C'_n and a copy of C''_n, and has an input x for C', and input y for C'', and an extra input bit b. It outputs the string $(C'_n(x) \wedge b) \vee (C''_n(y) \wedge \bar{b})$ where the combination with b and $\bar{b}$ is done for each bit position. □

Note that in the above proof, the depth of the circuit for $A \cup B$ is two more than the maximum depth of the circuits for A and B. Since union is associative, a union of k sets can be expressed as a binary tree of unions of depth $\lceil \log k \rceil$. Thus the union of k languages, each with an NC^0 proof system of depth d, has an NC^0 proof system of depth $d + 2\lceil \log_2 k \rceil$. In particular, we get the following nonuniform upper bounds.

Lemma 3.11 *Let $L \subseteq \{0,1\}^*$ have the property that there is a constant k such that for each n, $|L \cap \{0,1\}^n| \leq k$. That is, at each length, at most k strings of that length are in L. Then L admits a nonuniform NC^0 proof system.*

In certain cases, the complement of a language with an NC^0 proof system also has an NC^0 proof system. For example:

Lemma 3.12 *Let $L \subseteq \{0,1\}^*$ have the property that there is a constant k such that for each n, $|L \cap \{0,1\}^n| \geq 2^n - k$. That is, at each length, at most k strings of that length are not in L. Then L admits a nonuniform NC^0 proof system.*

Proof. The circuit C for $\mathrm{OR}^{-1}(1)$ outputs all strings except the string of all 0s. We first generalize this to exclude any fixed string y from the output. This is done as follows: Let $y \in \{0,1\}^n$ be the string that is to be excluded from the output of our proof circuit. Take the output bits $w_1, \ldots, w_n$ of C and feed them to a layer of XOR gates that does a bit-by-bit XOR of w and y. The output of the XOR layer is our output string. Since C never outputs all 0s, the output after XOR-ing with y can never be y.

Now we push this further to exclude k strings.

Let $L^{=n} = \{0,1\}^n \setminus U$, where $U = \{u^1, u^2, \ldots, u^k\}$ and $u^1, \ldots, u^k \in \{0,1\}^n$ are the strings excluded from L.

The proof is by induction on $|U|$. The base case of $|U| = 1$ has already been shown.

Assume we have a proof circuit for $L \setminus U$ for all U with $|U| < k$. Induction step: $|U| = k$. Let l be the first position where there is at least one string in U which has 0 at l and at least one string in U that has a 1 at l. Since $|U| > 1$, there exists such an l. Now partition U into U^0 and U^1 based on whether a string has a 0 or a 1 at the l'th position. Now by the choice of l, $|U^0| < k$ and $|U^1| < k$. From the induction hypothesis we have a proof circuit C^0 for $L \setminus U^0$

and a proof circuit C^1 for $L \setminus U^1$. We construct proof circuit C for L that takes k bits as input and outputs n bits as follows: Let $s \in L$ be an arbitrary fixed string. Define $C(bx)$ where b is a bit as follows:

- $C(bx) = C^0(x)$ if $b = 0$ and $C^0(x)_l = 0$;
- $C(bx) = C^1(x)$ if $b = 1$ and $C^1(x)_l = 1$;
- $C(bx) = s$ otherwise. □

Proposition 3.13 *Every language decidable in nonuniform* NC^0 *has a nonuniform* NC^0 *proof system.*

Proof. If the circuit C accepts a word w, then let D be the circuit extending C, which outputs the input if C accepts and otherwise outputs w. Then D enumerates the words accepted by C. □

4 Lower bounds

We now consider languages which do not admit NC^0 proof systems, some of them even regular. At first we focus on non-constant lower bounds for the depth required in order to enumerate these languages by circuits with binary gates. Later on we take the opposite perspective and ask, given a constant depth bound d, how large a fraction of a language can be enumerated by an NC^0 proof system of depth d. This fraction can turn out to be exponentially small. All our examples in this section are characterized by some counting feature.

4.1 Lower bounds on depth

We begin with our main concrete example of a non-NC^0-enumerable language.

Theorem 4.1 *The function Exact-OR*^{n} *on n bits, that evaluates to 1 if and only if exactly one of the input bits is 1, does not admit* NC^0 *proof systems.*

Proof. Suppose there is such a proof system, namely an NC^0-computable function $f : \{0,1\}^m \longrightarrow \{0,1\}^n$. Let $R_i \subseteq [m]$ be the proof bit positions that have a path to the ith output bit. For each i, there is at least one setting of the R_i bits that places a 1 in the ith bit of the output (producing the output string e_i). All extensions of this setting must produce e_i. Therefore $|f^{-1}(e_i)| \geq 2^{m-|R_i|}$. Let $c = \max_{i=1}^n |R_i|$; by assumption, $c \in O(1)$. Then for each $i \in [n]$, $|f^{-1}(e_i)| \geq 2^{m-c}$. But the $f^{-1}(e_i)$ partition $\{0,1\}^m$. Hence

$$2^m = \sum_{i=1}^{n} |f^{-1}(e_i)| \geq \sum_{i=1}^{n} 2^{m-c} = n2^{m-c}.$$

Therefore $c \geq \log n$, so $\exists i \in [n] : |R_i| \geq \log n$, a contradiction. □

Generalising this proof technique, we derive below a criterion which implies non-constant lower bounds for the depth of an enumerating circuit family.

Theorem 4.2 *Let L be a language and $\ell, t\colon \mathbb{N} \to \mathbb{N}$ functions. Suppose for each length n where $L^{=n}$ is non-empty, there is a set W of $t(n)$ distinct strings $W = \{w_1, \ldots, w_{t(n)}\} \subseteq L^{=n}$ satisfying the following: For each $w \in W$, there exists a set $S = \{i_1, \ldots, i_{\ell(n)}\} \subseteq [n]$ of $\ell(n)$ positions such that if x is in $L^{=n}$, and if x agrees with w on S, then x equals w. That is, the bits of w in positions indexed by S fix all the remaining bits. Then the depth of any bounded fan-in circuit family that enumerates L is at least* $\log\log t(n) - \log \ell(n)$.

Proof. Let $f : \{0,1\}^{m(n)} \longrightarrow \{0,1\}^n$ be a depth-$d(n)$-circuit enumerating the length n members of L, and let $\ell(n)$ and $t(n)$ be as in the statement of the theorem. Denote the resulting words $w_1, \ldots, w_{t(n)}$.

For each of the w_j the following holds: The $\ell(n)$ crucial bits have paths to at most $r(n) = \ell(n)2^{d(n)}$ bits of the proof. Thus there is a setting to $r(n)$ bits of the proof, all extensions of which generate the same output w_j. Hence $|f^{-1}(w_j)| \geq 2^{m(n)-r(n)}$.

Now we just count the number of proofs. As there are $m(n)$ proof bits,

$$2^{m(n)} = \text{number of proofs} \geq \sum_{j=1}^{t(n)} \text{number of proofs for } w_j \geq t(n)2^{m(n)-r(n)}$$

and hence

$$2^{r(n)} \geq t(n); \quad \ell(n)2^{d(n)} = r(n) \geq \log t(n); \quad d(n) \geq \log\log t(n) - \log \ell(n).$$

Using this theorem, we can show that several functions are not enumerable in constant depth.

Exact Counting. Consider the function Exact-Count^n_k on n bits: it evaluates to 1 if and only if exactly k of the input bits are 1. (Exact-OR^n is precisely Exact-Count$^n_1$.) For each length n there are exactly $\binom{n}{k}$ words in Exact-Count^n_k. And whenever k bits of a word are set to value 1, then all remaining bits are bound to take the value 0. So for Exact-Count^n_k the parameters $t(n)$ and $\ell(n)$ defined in the theorem above take the values $\binom{n}{k}$ and k, respectively, which yields a lower bound of

$$d(n) = \log\log \binom{n}{k} - \log k \geq \log\log \left(\frac{n^k}{k^k}\right) - \log k = \log(\log n - \log k)$$

on the depth of an enumerating circuit family. For $k(n)$ sub-linear in n this gives an unbounded function; thus for every sub-linear $k(n)$, Exact-Count^n_k does not admit an NC^0 proof system. Note that for a constant k, this language is even regular.

The threshold functions $\neg\mathbf{Th}^n_{k+1}$ and dually $\mathbf{Th}^n_{n-k}$ for sub-linear k. Let Th^n_a be the function that evaluates to 1, if and only if at least a of the n inputs are set to 1. The lower bounds for these languages are derived precisely by the same argument given above for Exact-Count^n_k. So they also yield the same set of parameters.

In more detail: Strings in Th^n_{n-k} (or $\neg\mathrm{Th}^n_{k+1}$) can have at most k 0s (at most k 1s, respectively). There are $t(n) = \binom{n}{k}$ ways of choosing $l(n) = k$ positions from $[n]$; for each such choice, setting the bits in the chosen positions to 0 (1, resp.) forces all other bits to be 1 (0, resp.).

The language 0^*1^* and iterations. First consider 0^*1^*, whose members consist of a (possibly empty) block of 0's followed by a (possibly empty) block of 1's. The $n+1$ length-n members of 0^*1^* are in 1-1 correspondence to the members of Exact-Count$^{n+1}_1$ via the NC^0 mapping $w_1 \dots w_n \longmapsto x_1 \dots x_{n+1}$, where $x_i := w_{i-1} \oplus w_i$, with the convention that $w_0 := 0$ and $w_{n+1} := 1$. Thus an NC^0 proof system of 0^*1^* would directly yield one for Exact-Count$^{n+1}_1$, which we have shown to be impossible. The parameters from the theorem are $\ell(n) = 2$ (two consecutive bits with different values or simply $w_1 = 1$ or $w_n = 0$) and $t(n) = n + 1$. By the same argument, for sub-linear k, the languages consisting of either exactly or up to k alternating blocks of 0's and 1's do not admit NC^0 proof systems.

Majority. The Majority language consists of those words which have at least as many 1's as 0's. Majority also does not admit an NC^0 proof system. But this does not follow from an extension of the techniques described so far, and requires a completely different and significantly more non-trivial approach. Instead of presenting it here, we devote the entire next section (Section 5) to this proof.

4.2 List enumerations

Consider a circuit $C\colon \{0,1\}^m \longrightarrow \{0,1\}^{tn}$. On input x, C can be thought of as producing a list $L(x)$ of t strings of length n. (An alternative view is that we allow t circuits, here merged into one, to enumerate words of length n.) We say that C *t-enumerates* L or is a *t-list proof system for* L if $\bigcup_x L(x) = L$. All along what we have been considering is $t = 1$.

For instance, every sparse language admits a nonuniform NC^0 polynomial-list proof system, as every word can be generated by a sub-circuit with constant output. So in particular, the regular languages Exact-Count^n_k for constant k are of this kind, though they do not have NC^0 proof systems. We observe below that any sub-language of Exact-Count$^n_1$ enumerated by a single circuit is small, and hence Exact-Count$^n_1$ **requires** $\Omega(n)$-lists. We will use this in Theorem 4.7 to prove a lower bound for the list length of the language of all permutation matrices.

Lemma 4.3 *Let L be a subset of Exact-Count$_k^n$ that has an NC^0 proof system which is computed by a depth d circuit family. Then for each length n the set $L^{=n}$ of length n members of L has at most 2^{k2^d} elements.*

Proof. This follows directly from Theorem 4.2, replacing $t(n)$ by $|L^{=n}|$. □

The Sunflower Lemma of Erdős-Rado gives rise to an alternative proof of a variant of the last lemma, albeit with a considerably weaker upper bound on the size of the enumerated fraction, e.g., for $k = 1$ the upper bound is $2^d!2^{2^{2d}}$. Here, a sunflower is formed out of sets of input bit positions that influence the relevant subsets of output bits. For completeness, we describe this proof technique below for $k = 1$. Unfortunately, even this technique does not yield a lower bound for Majority.

Proposition 4.4 (Erdős-Rado) *Let $\mathcal{F} = \{S_1, S_2, \ldots, S_k\}$ be a family of subsets of some universe. If*

1. *$|S_i| \leq \ell$ for all i, and*
2. *$k > (p-1)^l \cdot l!$,*

then $\mathcal{F}$ contains a sunflower with p petals. That is, there is a subfamily $\mathcal{F}' = \{T_1, \ldots, T_p\}$ with each $T_i \in \mathcal{F}$ such that for some set T, for any two $i \neq j$, $T_i \cap T_j = T$. T is called the core of the sunflower.

Lemma 4.5 *Let $C : \{0,1\}^m \longrightarrow \{0,1\}^n$ be a depth d circuit whose outputs are a subset of Exact-Count$_1^n$. Then C produces at most $N = (2^d)!2^{2^{2d}}$ distinct outputs.*

Proof. We prove this by contradiction. Assume that C enumerates more than N distinct outputs.

Let R_i denote the bits of the input that reach the ith bit of the output. Each R_i is of size at most 2^d. Set $\mathcal{F} = \{R_i \mid C \text{ can produce a 1 in position } i\}$. By our assumption, $|\mathcal{F}| > N$.

Set $\ell = 2^d$, $p = 2^{2^d} + 1$. Then $|\mathcal{F}| > (p-1)^\ell \cdot \ell!$, so there is a sunflower with p petals. Let the indices of the sets forming the petals be $I \subseteq [n]$, $|I| = p$. Let R be the core; clearly $|R| \leq 2^d$. For distinct $i, j \in I$, $R_i \cap R_j = R$.

Now consider any setting α to the proof bits in R. If there is an extension that produces a 1 in a position i indexed by I, then all extensions of α must produce zeroes in positions $j \in I \backslash \{i\}$. So no core setting can have two extensions with 1s in two different positions. However, there are (distinct) core settings that can produce a 1 in each of the petals. Thus

$$\#\text{petals} \leq \#\text{number of core settings} \leq 2^{|R|}$$

that is, $2^{2^d} + 1 \leq 2^{2^d}$, a contradiction. □

Remark 4.6 A simple modification allows the proof to go through for $\neg\mathrm{Th}_2^n$ as well. That is, a depth-d circuit can produce at most N outputs from Exact-OR, in addition to possibly the all-zeroes string. (The argument above only uses the fact that two 1s are disallowed, not that one 1 is required.)

A permutation matrix of order n is an $n \times n$ 0-1-matrix in which every row and every column contains exactly one 1. Lemma 4.3 or Lemma 4.5 give the following:

Theorem 4.7 *If C is a depth d circuit that t-enumerates the set of all permutation matrices of order n, then t grows exponentially with n.*

Proof. The circuit C can be thought of as t distinct circuits $C_1, \ldots, C_t$ with the same proof. Each row of each matrix output by each C_i belongs to Exact-OR. By Lemma 4.3, each C_i can construct at most $(2^{2^d})^n$ matrices (it has at most 2^{2^d} choices for each row). But the total number of choices must be at least the number of permutation matrices. Thus $n! \leq t(2^{2^d})^n$. Hence if $d \in O(1)$, t must be exponentially large. □

The same idea also works for proving lower bounds on the list length of enumerations of matrices which encode all Hamiltonian cycles in a complete graph or all paths from 1 to n in K_n.

4.3 Constant influence

Motivated by their investigation into NC^0 cryptography [5, 6], Applebaum et al. [7] investigate cryptography with constant input locality. As a related question we ask which languages can be proven by circuits that have the property that every input bit influences only constantly many output bits.

In the remainder of this section, we look at proof circuits that are NC^0 like before, but with the added restriction that each input bit can only influence constantly many output bits. We show that Exact-Count$_k^n$ and Th_k^n do not have proof circuits with this added restriction for suitable values of n and k.

The proof is based on the following observation for any proof circuit for Exact-Count$_k^n$: Let C be a proof circuit for Exact-Count$_k^n$. For an output gate i of C we denote by $\sup(i)$ the set of all input gates of C that have a path to i. For a set S of output gates of C we let $\sup(S) = \bigcup_{i \in S} \sup(i)$. Now, for any set of output positions $S \subseteq [n]$, $|S| = k$ and $i \in [n], i \notin S$, we have $\sup(S) \cap \sup(i) \neq \emptyset$. If this were not true, then we could obtain $(k+1)$ 1s in the output by setting the bits in $\sup(S)$ to get k 1s corresponding to the positions in S, and by setting the $\sup(i)$ to get a 1 in the ith output position.

The above can be generalized to the following:

Lemma 4.8 *The language Exact-Count$_k^n$ does not have a proof circuit of depth d with each input bit influencing at most c output bits if $n \geq c2^d + k$.*

Proof. Suppose such a circuit exists, take any output position $i \in [n]$. We know that $|\sup(i)| \leq 2^d$. Let T be the set of all output bits j for which $\sup(i) \cap$

$\sup(j) \neq \emptyset$. $|T| \leq c2^d$. Now if $n \geq c2^d + k$, then we can find a set $S \subseteq [n]$ of output positions such that $|S| = k$ and $S \cap T = \emptyset$. This implies that $\sup(S) \cap \sup(i) = \emptyset$, contradicting the observation made above. □

Corollary 4.9 *The language Exact-Count$^n_{n/2}$ does not have a proof system of constant depth and constant influence.*

A similar observation as above holds for threshold functions as well: Let C be a proof circuit for Th^n_k. Then, for any subset of output positions $S \subseteq [n]$, $|S| = n - k$ and any $i \notin [n]$, we have $\sup(S) \cap \sup(i) \neq \emptyset$. If this were not true, then we can force C to output $n - k + 1$ 0s by setting the support of S and the support of i such that we get 0s in all the S positions and position i.

Lemma 4.10 *The function Th^n_k does not have a proof circuit of depth d with each input bit influencing at most c output bits if $n > k \geq c2^d$.*

Proof. Suppose such a circuit exists, call it C. Take any output position $i \in [n]$. We know that $|\sup(i)| \leq 2^d$. Let T be the set of all output bits which have a support bit in $\sup(i)$. $|T| \leq c2^d$. Now since $k \geq c2^d$, we can find a set of output positions $S \subseteq [n]$ with $|S| = n - k$ such that $S \cap T = \emptyset$. Since T was all the bits that are influenced by $\sup(i)$, we have $\sup(S) \cap \sup(i) = \emptyset$. The above observation can be used to conclude that C can be forced to output a string that has more than $n - k$ 0s. □

The following is an easy corollary; it is, however, subsumed by the much stronger (and much more difficult to prove) Theorem 5.1 proved in the next section.

Corollary 4.11 *Majority does not have a proof circuit family with constant depth and constant influence.*

5 Majority does not admit NC^0 proof systems

The language MAJ consists of all 0-1-words that contain more 1's than 0's. The language EXMAJ consists of all 0-1-words w that contain exactly $1 + \lfloor |w|/2 \rfloor$ 1's. Clearly, EXMAJ $\subseteq$ MAJ. If w is in EXMAJ, and if a single bit in w is flipped *from 1 to 0*, then the resulting string w' is not in MAJ. We will exploit this to show that MAJ does not admit an NC^0 proof system.

Intuitive Idea

Assume that there is an NC^0 proof system for MAJ. The idea of the proof is that there are two types of inputs: inputs that influence a linear number of outputs – call these the high-fanout inputs, and inputs that influence a sublinear number of outputs – these are the low-fanout inputs. (Note our non-standard use of the term *fanout* which refers to the number of output bits an input is connected to instead of the number of wires leaving the gate.) Since every output is

connected to a constant number of inputs, there can only be a constant number of high-fanout inputs. So nearly all inputs are of low-fanout.

We will try to find an output x_i whose value can be changed by only manipulating the set S of low-fanout inputs connected to x_i. Also, since low-fanout inputs are only connected to a sublinear number of outputs, we can assume that S is connected to less than $n/2$ of the outputs. So we can find a word w in MAJ that has a 1 at every position that depends on the input bits of S and assign the remaining outputs in such a way that we even get a word w in EXMAJ. Since this is a valid word in MAJ, the proof system needs to generate it, hence there is an assignment of the inputs that outputs w.

But now we can modify the input bits in S and toggle x_i to the value 0. Toggling the input bits in S only affects output bits that were assigned to 1, hence this might flip additional bits from value 1 to value 0. But the word generated in this way by the proof system has fewer 1's than w and hence is not in MAJ. It follows there is no NC^0 proof system for MAJ.

Formalising this idea is a bit more complicated. It turns out that we need a finer gradation of what we consider high-fanout. We will define a decreasing function $g : \mathbb{N} \longrightarrow \mathbb{N}$, and at stage e, we consider an input connected to more than $g(e)$ output bits as high-fanout. Say that X_e is the set of high-fanout inputs at stage e. If we can find an output x_i as above, we will have obtained a contradiction. But we may not immediately succeed in finding such an x_i, because it may be the case that settings to the high-fanout bits X_e fix each output x_i. We then carefully fix a small set R_e of output bits and an assignment w^e to these bits in a way such that each output outside of R_e can be toggled without changing the input setting to X_e. At this point, we look for a string in EXMAJ agreeing with w^e, and try to obtain a contradiction by toggling a carefully chosen output. If we still cannot obtain a contradiction, we move on to the next stage. Finally, we show that if we complete stage c for some suitably chosen constant c, then we get a different kind of contradiction: a few high-fanout input bits completely determine many output bits. A simple counting argument shows that this cannot happen for MAJ.

To make this argument rigorous, we define a certain assertion Π_e concerning stage e. This assertion states that there is a setting w^e to a set R_e of output bits satisfying 4 properties: (1) R_e is small, (2) assignments to X_e compatible with w^e do not fix any output bit outside R_e, (3) the forbidden set F_{e+1}, consisting of output bits sharing a low-fanout input with some bit in R_e, is small, and (4) every non-forbidden-output is connected to at least one input that will enter the high-fanout set at stage $e + 1$. Then the above argument can be rephrased as: Π_0 is true, $\Pi_e \Rightarrow \Pi_{e+1}$, but at least one of $\Pi_0, \ldots, \Pi_{c-1}$ is not true. This is obviously a contradiction.

With this idea in mind, we now state and prove our theorem.

Theorem 5.1 *The language* MAJ *of all 0-1-words that contain more 1's than 0's does not admit an* NC^0 *proof system.*

Proof. Assume that (C_n) is an NC^0 family of circuits enumerating MAJ. Let d be the maximal depth of the circuits C_n and let $c \leq 2^d$ be the maximum

number of input bits connected to the same output bit. It is easy to see that no projection can be a proof system for MAJ. Hence $c \geq 2$.

Let In and Out denote the sets of input and output bits of C_n, respectively. For a set A of nodes of the circuit, define the sets

$$\begin{aligned} \mathrm{Out}(A) &:= \{y \in \mathrm{Out} \mid y \text{ is connected to some } x \in A\} \\ \mathrm{In}(A) &:= \{x \in \mathrm{In} \mid x \text{ is connected to some } y \in A\} \end{aligned}$$

For a singleton $\{x\}$ whose only element is an input/output bit, we simply write $\mathrm{Out}(x)$ or $\mathrm{In}(x)$.

Define functions f, g as follows:

$$\begin{aligned} f(e) &:= \begin{cases} 1 & \text{for } e = 0 \\ c^{5f(e-1)+1} & \text{for } e > 0 \end{cases} \\ g(e) &:= \frac{cn}{f(e)} \end{aligned}$$

Clearly, f is an increasing function, and g is a decreasing function. Note that f does not depend on the value of n. All arguments in the proof will work for a choice of $n \geq 4 \cdot f(c)$. We use g to define the high-fanout set at each stage;

$$X_e := \left\{ x \in \mathrm{In} \;\middle|\; |\mathrm{Out}(x)| > g(e) \right\}$$

Note that for each e, $X_{e-1} \subseteq X_e$. Also, since there are at most cn input-output-connections in circuit C_n, and since each input bit in X_e contributes more than $g(e)$ input-output connections, we obtain

$$|X_e| \cdot g(e) < (\text{number of input-output connections in } C_n) \leq cn = f(e) \cdot g(e).$$

Thus the function $f(e)$ yields an upper bound for the size of X_e.

We now state an assertion concerning the circuit C_n, for a parameter e; call this assertion Π_e.

Assertion (Π_e) *There exists a set $R_e \subseteq \mathrm{Out}$, and a setting w^e to R_e, satisfying the following properties.*

1. $|R_e| \leq 2^{f(e)+1}$.
2. *for each $y \in \mathrm{Out} \setminus R_e$, for each assignment $q : X_e \to \{0,1\}$ compatible with w^e, and for each value $b \in \{0,1\}$, there is a legal configuration of the circuit extending $w^e \cup q$ and setting y to b.*
3. *Let $F_{e+1} := R_e \cup \mathrm{Out}(\mathrm{In}(R_e) \setminus X_{e+1})$. ($F_{e+1}$ denotes the set of* forbidden *outputs.) Then $|F_{e+1}| \leq \frac{n}{c^3}$.*
4. $\forall y \in \mathrm{Out} \setminus F_{e+1}$, $\mathrm{In}(y) \cap (X_{e+1} \setminus X_e) \neq \emptyset$.

We prove the theorem by contradiction. Assuming that (C_n) enumerates MAJ, we will show that for all sufficiently large n, the following statements hold:

(A). Π_0 is true.

(B). $\Pi_0, \Pi_1, \ldots, \Pi_{c-1}$ are not simultaneously true.

(C). For all $1 \leq e < c$, $\Pi_{e-1} \implies \Pi_e$.

With these statements established in Lemmas 5.2, 5.3, and 5.4 below, we reach a contradiction, and the proof of the main theorem is complete. □

Proof of the Statements (A)-(C)

Lemma 5.2 (Statement (A)) Π_0 *is true.*

Proof. Note that $X_0 = \emptyset$. Define $R_0 = \emptyset$, $w^0 = \epsilon$. Then $F_1 = \emptyset$. Properties 1,3 are trivial. Property 2 holds because no bit in MAJ is fixed; each y can take values 0 and 1.

It remains to show Property 4. Suppose Property 4 fails; that is, there is an output bit y with no connections to X_1. Then the neighbourhood of y, defined as $N(y) = \text{Out}(\text{In}(y))$, has size at most $c \times g(1) = n/c^4 < n/2$. So there exists a string z in EXMAJ with only 1s at members of $N(y)$, and hence a configuration β of the circuit compatible with z. By changing the input settings in β only in $\text{In}(y)$, we can set output y to 0. Since $\text{In}(y)$ reached only positions set to 1 in β, the change strictly decreases the number of 1s in the output. Thus C_n outputs a string not in MAJ, a contradiction. Hence Property 4 must hold. □

Lemma 5.3 (Statement (B)) $\Pi_0, \Pi_1, \ldots, \Pi_{c-1}$ *are not simultaneously true.*

Proof. Assume to the contrary that for each $e \in \{0, 1, \ldots, c-1\}$, Π_e is true. Define $F = \cup_{e=1}^{c} F_e$, and let $G := \text{Out} \setminus F$ denote the remaining output bits.

Consider any output bit $y \in G$. For each $e \in [c]$, y is not in F_e, so by property 4 in Π_{e-1}, $\text{In}(y)$ has a bit in $X_e \setminus X_{e-1}$. Thus $\text{In}(y)$ has at least c bits in X_c. But $\text{In}(y)$ has at most c bits overall, so $\text{In}(y)$ is in fact completely contained in X_c.

By property 3 of each Π_e, we know that $|G| \geq n - c(n/c^3) = n - n/c^2 \geq 3n/4$. As remarked earlier, $f(e)$ is an upper bound on $|X_e|$, and so $|X_c| < f(c)$. We saw above that for each $y \in G$, $\text{In}(y) \subseteq X_c$. Thus, in legal configurations of the circuit, the assignment to G is determined by the assignment to X_c. But there are less than $2^{f(c)}$ distinct assignments to X_c, while there are at least $\binom{|G|}{\frac{n}{4}} \geq 2^{n/4}$ distinct assignments to G corresponding to strings in MAJ. For sufficiently large n, this is impossible. □

Lemma 5.4 (Statement (C)) *For all* $1 \leq e < c$, $\Pi_{e-1} \implies \Pi_e$.

Proof. To show that Π_e holds, we first describe a procedure that extends R_{e-1} and w^{e-1} to R_e and w_e, and then show that the extension satisfies properties 1 to 4 of Π_e. The immediate objective of the extension procedure is to satisfy property 2 of Π_e; control the input bits in $X_e \setminus X_{e-1}$ by restricting the output to a configuration that does not allow the X_e part of the input to fix further output bits.

Set $R = R_{e-1}$ and $w = w^{e-1}$. Define the set Q as follows.

$$Q := \left\{ q \in \{0,1\}^{X_e} \;\middle|\; q \text{ is compatible with } w \right\}.$$

Perform the Prune procedure described below.

The Prune Procedure: Perform the following step as long as possible.

> Find a partial configuration $q \in Q$, a position y in $\mathrm{Out} \setminus R$, and a value $b \in \{0,1\}$ such that all configurations of the circuit extending q set y to b. Add y to R, set y to $\overline{b}$ in w making w incompatible with q, and remove from Q all assignments (including q) that are incompatible with w.

After the Prune procedure terminates, set R_e to the resulting R, and w^e to the resulting w.

The four claims below show that this choice of R_e and w^e satisfies Π_e. First, we state a simple but important observation: After every step in the Prune procedure, Q satisfies

$$Q = \left\{ q \in \{0,1\}^{X_e} \;\middle|\; q \text{ is compatible with } w \right\}.$$

In connection with the first property stated in Π_e and proven below this implies that Q is not empty as long as $n > 2^{f(e)+2}$.

Claim 5.5 (Property 1 of Π_e holds) $|R_e| \leq 2^{f(e)+1}$.

Proof. We start with $|Q| = 2^{|X_e|}$ and $R = R_{e-1}$. Each time we add a position to R, we discard at least one element from Q. So $|R_e| \leq |R_{e-1}| + 2^{|X_e|}$. Using the fact that $f(e)$ is an upper bound for $|X_e|$, the property 1 of Π_{e-1}, and the definition of f, we get: $|R_e| \leq |R_{e-1}| + 2^{|X_e|} \leq 2^{f(e-1)+1} + 2^{f(e)} \leq 2^{f(e)+1}$. □

Claim 5.6 (Property 2 of Π_e holds) *For each $y \in \mathrm{Out} \setminus R_e$, for each assignment $q : X_e \to \{0,1\}$ compatible with w^e, and for each value $b \in \{0,1\}$, there is a legal configuration of the circuit extending $w^e \cup q$ and setting y to b.*

Proof. Recall that, the way the Prune procedure is defined, all settings $q : X_e \longrightarrow \{0,1\}$ compatible with w^e are in Q, and none of them determine the bit at any position $y \in \mathrm{Out} \setminus R_e$. Hence for any such y, and any value b, it is possible to extend $q \cup w^e$ and set the bit at position y to b. □

Claim 5.7 (Property 3 of Π_e holds) *For sufficiently large n, $|F_{e+1}| \leq n/c^3$.*

Proof. Recall that $F_{e+1} := R_e \cup \mathrm{Out}(\mathrm{In}(R_e) \setminus X_{e+1})$.

$$\begin{aligned}
\frac{n}{c^3} - |F_{e+1}| &\geq \frac{n}{c^3} - (|R_e| + c \cdot |R_e| \cdot g(e+1)) \\
&= \frac{n}{c^3} - |R_e| \left(1 + c \cdot \frac{nc}{f(e+1)}\right) \\
&\geq \frac{n}{c^3} - 2^{f(e)+1} \cdot \left(1 + \frac{nc^2}{f(e+1)}\right) \quad \text{using Claim 5.5} \\
&= n \cdot \left(\frac{1}{c^3} - \frac{c^2 2^{f(e)+1}}{f(e+1)}\right) - 2^{f(e)+1} \\
&= \delta_e n - 2^{f(e)+1}
\end{aligned}$$

It suffices to show that $\delta_e > 0$, because then we can choose a sufficiently large n and ensure that $\delta_e n$ exceeds $2^{f(e)+1}$. (Note, for $e \leq c$, δ_e and $f(e)$ are constants independent of n.)

$$\begin{aligned}
\delta_e > 0 &\iff \delta_e c^3 f(e+1) > 0. \\
\delta_e c^3 f(e+1) &= f(e+1) - c^5 2^{f(e)+1} \\
&= c^{5f(e)+1} - c^5 2^{f(e)+1} \\
&\geq 2^{f(e)+1} c^{4f(e)} - c^5 2^{f(e)+1} \qquad \text{since } c \geq 2 \\
&= 2^{f(e)+1} \left[c^{4f(e)} - c^5\right] \\
&\geq 0 \qquad \text{since } e \geq 1 \text{ and } c \geq 2,\ 4f(e) \geq 5. \qquad \square
\end{aligned}$$

Proof (Alternative proof.). Using Claim 1, the definitions of X_{e+1}, g and f, and the facts that $c \geq 2$ and $1 \leq e \leq c-1$ along with the choice of $n \geq 4f(c)$ we get

$$\begin{aligned}
|F_{e+1}| &\leq |R_e| + |\,\mathrm{Out}(\mathrm{In}(R_e) \setminus X_{e+1})| \\
&\leq 2^{f(e)+1} + g(e+1) \cdot 2^{f(e)+1} \cdot c \\
&= 2^{f(e)+1} + \frac{cn}{f(e+1)} \cdot 2^{f(e)+1} \cdot c \\
&\leq c^{f(c-1)+1} + \frac{c^2 n 2^{f(e)+1}}{c^{5f(e)+1}} \\
&\leq \frac{f(c)}{c^4} + \frac{c^2 n}{c^6} \cdot \frac{2^{f(e)+1}}{c^{f(e)+1}} \\
&\leq \frac{n}{c^4} + \frac{n}{c^4} = \frac{n}{c^3} \qquad \square
\end{aligned}$$

Claim 5.8 (Property 4 of Π_e holds) *For sufficiently large n, $\forall y \in \mathrm{Out} \setminus F_{e+1}$, $\mathrm{In}(y) \cap (X_{e+1} \setminus X_e) \neq \emptyset$.*

Proof. Suppose the claim does not hold. That is, suppose there exists a $y \in \mathrm{Out} \setminus F_{e+1}$ such that $\mathrm{In}(y) \cap (X_{e+1} \setminus X_e) = \emptyset$. But note that $\mathrm{In}(y) \cap \mathrm{In}(R_e) \subseteq X_{e+1}$; otherwise y would have been in F_{e+1} by definition. Putting these together,

we conclude that $\mathrm{In}(y) \cap \mathrm{In}(R_e) \subseteq X_e$. Generalising the corresponding argument used in establishing statement (A), we will now show that this is not possible.

Consider the e-neighbourhood of y defined as $U := \mathrm{Out}(\mathrm{In}(y) \setminus X_e)$. Since $y \notin F_{e+1}$, the sets U and R_e are disjoint. We have that $|U| \leq n/c^4$, since by definition of f, for $e > 0$, $f(e) \geq c^6$, and hence $|U| \leq c \cdot g(e) \leq c^2 n/f(e) \leq n/c^4$. Together with Claim 5.5, for sufficiently large n, $|R_e \cup U| < n/2$. Hence there exists a string z in ExMaj with only 1s at positions in U, and according to w^e at positions in R_e. Hence there is a configuration β of the circuit compatible with z; let this configuration restricted to X_e be α. (Thus β extends $\alpha \cup w^e$.) We have already established property 2 of Π_e. Applying this to y and α with $b = 0$, we conclude that there is another configuration γ, also extending $\alpha \cup w^e$, such that γ has a 0 at y.

Now change the input settings of β only at positions in $\mathrm{In}(y) \setminus X_e$ to match the settings in γ. This change can affect only the output bits in U. In particular, it changes output y from 1 to 0. Since U had only 1s in β, the change strictly decreases the number of 1s in the output. Thus C_n outputs a string not in Maj, a contradiction. □

With Claims 5.5, 5.6, 5.7, 5.8, Lemma 5.4 is established. □

6 Proof systems for regular languages

In this section, we describe some sufficient conditions under which regular languages have NC^0 proof systems. The regular languages we consider may not necessarily be over a binary alphabet, but we assume that a binary (letter-by-letter) encoding is output.

Our first sufficient condition abstracts the strategy used to show that OR has an NC^0 proof system. This strategy exploits the fact that there is a DFA for OR, where every useful state has a path to an "absorbing" final state. (Here, by useful we mean that the state q lies on some path from the start state to a final state. This is syntactic usefulness, and may not correspond to real usefulness if for each such path through q there is also another accepting path avoiding q.)

Theorem 6.1 *Let L be a regular language accepted by an NFA $M = (Q, \Sigma, \delta, F, q_0)$. Let $F' \subseteq F$ denote the set of absorbing final states; that is, $F' = \{f \in F \mid \forall a \in \Sigma, \delta(f, a) = f\}$. Suppose M satisfies the following condition:*

> *For each $q \in Q$, if there is a path from q to some $f \in F$, then there is a path from q to some $f' \in F'$.*

Then L has an NC^0 proof system.

Proof. We assume without loss of generality that all states of M are useful (they lie on some accepting path); if not, we remove non-useful states (and the hypothesised property continues to hold). The hypothesis is that from each state q, we can reach some absorbing final state via a word of length at most $k = |Q| - 1$. Pick any such word arbitrarily, pad it arbitrarily with a suffix so

that its length is exactly k, and denote the resulting word as $\mathrm{fin}(q)$ (i.e., $\mathrm{fin}(q)$ "finalizes" q). Clearly, $\delta(q, \mathrm{fin}(q)) \in F'$.

The proof consists of the word x broken into blocks of size k, with the remainder bits at the beginning. In addition, the proof provides the state of M after each block on some accepting run. So the total proof is $x^0, x^1, \ldots, x^N, q^1, \ldots, q^N$ where $N = \lfloor n/k \rfloor$, each $q^i \in Q$, $x^i \in \Sigma^k$ for $i \geq 1$, and $x^0 \in \Sigma^{<k}$ are the remainder bits.

The word w output by the proof system on such a proof is also broken into blocks in the same way, and each block is defined as follows:

$$\begin{aligned} w^0 &= x^0 \\ w^1 &= \begin{cases} x^1 & \text{if } q^2 \in \delta(q^0, x^0 x^1) \\ \mathrm{fin}(\delta(q_0, x^0)) & \text{otherwise.} \end{cases} \\ \text{For } 2 \leq i \leq N, w^i &= \begin{cases} x^i & \text{if } q^i \in \delta(q^{i-1}, x^i) \\ \mathrm{fin}(q^{i-1}) & \text{otherwise.} \end{cases} \end{aligned}$$

Since $|Q|$ and $|\Sigma|$ are constant, the transition function δ can be implemented by a circuit of constant size. And since k is a constant, checking if $q^i \in \delta(q^{i-1}, x^i)$ can be done in NC^0. Thus the above can be implemented in NC^0. □

Observe that the OR and the Exact-OR are both star-free languages but the complements in the expression for OR are applied to the empty set, whereas those in Exact-OR are applied to non-empty sets. Based on this, we formulate and prove the following sufficient condition for a star-free regular language to have an NC^0 proof system.

Definition 6.2 *Strict star-free expressions over an alphabet Σ are exactly the expressions obtained as follows:*

1. *ϵ, a for each $a \in \Sigma$, $\Sigma^* = \bar{\emptyset}$ are strict star free.*
2. *If r and s are strict star free, so is $(r \cdot s)$.*
3. *If r and s are strict star free, so is $(r + s)$.*

Theorem 6.3 *Let r be a strict star-free expression describing a language $L = L(r)$. Then L admits an NC^0 proof system.*

Proof. We first note that in a regular expression, $\cdot$ distributes over $+$. Hence it is possible to repeatedly apply this rule of distributivity to arrive at an expression that is of the form $s_1 + s_2 + \cdots + s_k$, where each s_i is simply a concatenation without any $+$. So we assume that we have a strict star-free regular expression in this form.

Now, if we can show that each of the expressions s_i has an NC^0 proof system, then, we can use the fact that NC^0 proof systems are closed under finite union (Lemma 3.10).

The following claim shows that this is indeed true:

Claim 6.4 *Let L be a language recognized by a strict star-free expression s that does not have a $+$. Then L admits NC^0 proof systems.*

Proof. The expression s must be of the form $w_1\ \bar{\emptyset}\ w_2\ \bar{\emptyset}\ \ldots\ w_{k-1}\ \bar{\emptyset}\ w_k$, where $w_i \in \Sigma^+$ for $1 < i < k$ and $w_1, w_k \in \Sigma^*$. Let $s = w_1\ \bar{\emptyset}\ w_2\ \bar{\emptyset}\ \ldots\ w_{k-1}\ \bar{\emptyset}\ w_k$. Note that if $w_1 \neq \epsilon$, then we can hardwire w_1 to be the first $|w_1|$ symbols in the output of our proof circuit. Similarly w_k can be hardwired at the end. Now for the central $\bar{\emptyset}\ w_2\ \bar{\emptyset}\ w_3\ \ldots\ w_{k-1}\ \bar{\emptyset}$ part: Notice that any minimal DFA for this expression will have a self-absorbing final state to which all states have a path. Hence Theorem 6.1 implies that we have an NC^0 proof system for this language. Using this NC^0 proof system, and hardwiring w_1 and w_k as prefix and suffix respectively, we obtain an NC^0 proof system for L. □

Theorem 6.1 essentially characterizes functions like OR. On the other hand, the parity function, that has an NC^0 proof system, cannot be recognized by any DFA or NFA with an absorbing final state. The strategy used in constructing the proof system for parity exploits the fact that the underlying graph of the DFA for parity is strongly connected. In the following result, we abstract this property and prove that strong connectivity in an NFA recogniser is indeed sufficient for the language to admit an NC^0 proof system.

Theorem 6.5 *Let L be accepted by NFA $M = (Q, \Sigma, \delta, F, q_0)$. If the directed graph underlying M is strongly connected, then L admits an NC^0 proof system.*

Proof. We use the term "walk" to denote a path that is not necessarily simple, and "closed walk" to denote a walk that begins and ends at the same vertex. The proof system circuit construction we describe below is applied only for those lengths where L is non-empty.

The idea behind the NC^0 proof system we will construct here is as follows: We take as input a sequence of blocks of symbols $x^1, x^2, \ldots, x^k$, each of length l and as proof, we take the sequence of states $q^1, q^2, \ldots, q^k$ that M reaches after each of these blocks, on some accepting run. Now we make the circuit verify at the end of each block whether that part of the proof is valid. If it is valid, then we output the block as is. Otherwise, if some x^i does not take M from q^{i-1} to q^i, then we want to make our circuit output a string of length l that indeed makes M go from q^{i-1} to q^i. So we make our circuit output a string of symbols which will first take M from q^{i-1} to q_0, then from q_0 to q^i. To ensure that this length is indeed l, we sandwich in between a string of symbols that takes M on a closed walk from q_0 to q_0. We now proceed to formally prove that closed walks of the required length always exist, and that this can be done in NC^0.

Define the following set of non-negative integers:

$$T = \{\, \ell \mid \text{there is a closed walk through } q_0 \text{ of length exactly } \ell \,\}$$

Since M is strongly connected, we know that T is non-empty. Let g be the greatest common divisor of all the numbers in T. Note that though T is infinite, it has a finite subset T' whose gcd is g.

Choose a subset S of states as follows:

$$S = \{\, q \in Q \mid \text{there is a walk from } q_0 \text{ to } q \text{ whose length is } 0 \bmod g \,\}$$

Claim 6.6 *For every $p \in Q$, $\exists \ell_p, r_p \in \{0, 1, \ldots, g-1\}$ such that*

1. *the length of every path from q_0 to p is $\equiv \ell_p \pmod g$;*
2. *the length of every path from p to q_0 is $\equiv r_p \pmod g$.*

Proof. Let ℓ, ℓ' be the lengths of two q_0-to-p paths, and let r, r' be the lengths of two p-to-q_0 paths. Then there are closed walks through q_0 of length $\ell + r$, $\ell + r'$, $\ell' + r$, $\ell' + r'$, and so g must divide all these lengths. So $\ell = -r \pmod g = -r' \pmod g$, and $r = -\ell \pmod g = -\ell' \pmod g$. It follows that $\ell \equiv \ell' \pmod g$ and $r \equiv r' \pmod g$. □

From here onwards, for each $p \in Q$, by ℓ_p and r_p we mean the numbers as defined in the above claim.

Claim 6.7 *For every $p \in S$, $\ell_p = r_p = 0$.*

Proof. By the definition of S, we have $\ell_p = 0$. Suppose $r_p \neq 0$. Let w be a word taking M from p to q_0. Appending this to any word w' that takes M from q_0 to p gives a closed walk through q_0 whose length is $0 + r_p \neq 0 \pmod g$. This contradicts the fact that g is the gcd of numbers in T. □

Claim 6.8 *There is a constant c_0 such that for every $K \geq c_0$, there is a closed walk through q_0 of length exactly Kg.*

Proof. This follows from Lemma 6.9 below. □

Note that if $g = 1$, then every state is in S, and for every state p, $l_p = r_p = 0$. Claim 6.8 then asserts that there are closed walks through q_0 of every possible length exceeding c_0.

Let $K = |Q|$. Now set $t = \lfloor \frac{K-1}{g} \rfloor$ and $\ell = t \cdot g$. Then for every $p \in S$, there is a path from q_0 to p of length $t'g$ on word $\alpha(p)$, and a path from p to q_0 of length $t''g$ on word $\beta(p)$, where $0 \leq t', t'' \leq t$. ($\alpha(p)$ and $\beta(p)$ are not necessarily unique. We can arbitrarily pick any such string.)

If for all accepting states $f \in F$, $\ell_f \not\equiv n \pmod g$, then $L^{=n} = \emptyset$, and the circuit C_n is empty.

Otherwise, let $r = n \pmod g$. There is at least one final state f such that $\ell_f \equiv r \pmod g$. Thus there is at least one string of length $t'g + r$, with $0 \leq t' \leq t$, that takes M from q_0 to f.

We now construct a proof circuit $C : \Sigma^n \times Q^n \longrightarrow \Sigma^n$. We consider the inputs of the proof circuit to be divided into blocks. We choose the block size to be a multiple of g, with the possible exception of the last block. In particular, we choose block size $cg = (2t + c_0)g$. The last block is of size $c'g + r$ for some $0 \leq c' < c$.

Let $k = \lfloor n/cg \rfloor$. Now the total proof is $x^1, \ldots, x^k, x^{k+1}, q^1, \ldots, q^k, q^{k+1}$ where each $q^i \in Q$, $x^i \in \Sigma^{cg}$ for $i \leq k$, and $x^{k+1} \in \Sigma^{c'g+r}$ for some $0 \leq c' < c$.

The word w output by the proof system on such a proof is also broken into blocks in the same way, and each block is obtained as follows:

1. For $1 \leq i < k$, if $q^i \in \delta(q^{i-1}, x^i)$, then $w^i = x^i$. Otherwise, w^i is obtained by concatenating $\beta(q^{i-1})$, a word u such that $q_0 \in \delta(q_0, u)$, and $\alpha(q^i)$. We need $|u| = (c - t' - t'')g$, and we know that $(c - t' - t'') \geq c_0 g$, and hence Claim 6.8 guarantees that such a word u exists.

2. If $q^{k+1} \in \delta(q^{k-1}, x^k x^{k+1})$ and $q^{k+1} \in F$, then let $w^k w^{k+1} = x^k x^{k+1}$.

   Otherwise, let $w^k w^{k+1}$ have as suffix a string of length $t'g + r$ in L, where $0 \leq t' \leq t$. By the choice of t we know that such a string exists. This leaves a prefix of length $(cg + c'g + r) - (t'g + r) = (c + c' - t)g$ with $(c + c' - t) \geq c_0 g$. We insert here a word u such that u takes q_0 to q_0; by Claim 6.8, such a word exists. □

Lemma 6.9 (Folklore) *Let T be a set of positive integers with gcd g. There is a constant c_0 such that for every $K \geq c_0$, Kg can be generated as a non-negative integral combination of the integers in T.*

Proof. We prove the statement by induction on $|T|$. Let $T = \{m_1, m_2, \ldots, m_t\}$ be the given set.

Basis: If $t = 1$, then $g = m_1$ and $Kg = Km_1$, so set c_0 to 1.

Inductive Hypothesis: Assume the statement is true for all sets of size $t - 1$.

Inductive Step: T is a set of size t.

It suffices to prove the statement when $g = 1$; for larger g, let T' be the set $\{t/g \mid t \in T\}$. Then T' has gcd 1, and if we can generate all numbers beyond c_0 with T', then we can generate all Kg for $K \geq c_0$ with T. So now assume T has gcd 1.

Let g' denote the gcd of the subset R consisting of the first $t - 1$ numbers. If $g' = 1$, then, even without using the last number m_t, we are already done by induction. Otherwise, let $m = m_t$. Then the numbers g', m are co-prime (because gcd for T is 1). By induction, there is a constant c' such that using only numbers from R, we can generate $K'g'$ for any $K' \geq c'$. Set $c = (c' + m)g$. Consider any number $n \geq c$.

The numbers $0 < n - (c' + m - 1)g, n - (c' + m - 2)g, \ldots, n - (c' + 1)g, n - c'g$ all have different residues modulo m.
(If not, suppose for some $0 \leq i < j \leq m - 1$, $n - (c' + i)g \equiv n - (c' + j)g \pmod{m}$. Then $(j - i)g \equiv 0 \pmod{m}$, and so m must divide $(j - i)g$. Since $0 < j - i < m$, m does not divide $j - i$. But m is co-prime to g. Contradiction.)
So for some $0 \leq i < m$, and for some non-negative integer a, $n - (c' + i)g = am$. That is, $n = (c' + i)g + am$. By the induction hypothesis, $(c' + i)g$ can be generated using numbers in $R \subseteq T$. And $m \in T$. So n can be generated from T. □

Example 6.10 The following shows the construction of the proof circuit as in the proof of Theorem 6.5 for the regular language $L = (1^9 + 0^6)^*$. Consider the NFA for L shown in Figure 1. Using the same notation as in Theorem 6.5, we

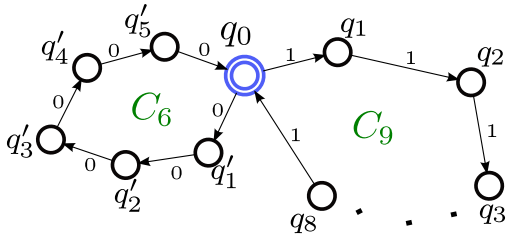

Figure 1: The strongly connected NFA from Example 6.10

have $T = \{6, 9\}$. The greatest common divisor of the numbers in T is $g = 3$. Then we have $S = \{q_3, q_6, q'_3\}$. It is easy to see that in our example, Claim 6.8 goes through for $c_0 = 2$. We choose the block length to be

$$\ell = \left\lfloor \frac{|Q| - 1}{g} \right\rfloor g + c_0 g + \left\lfloor \frac{|Q| - 1}{g} \right\rfloor g = 12 + 6 + 12 = 30.$$

This is chosen such that in the case of an input block that claims to take state p to state p' and does not have the correct proof, our proof system can output a set of at most 12 symbols to go from p to q_0, and then cycle around q_0 using kg symbols where $k > c_0$ is chosen appropriately and finally output at most 12 symbols to go from q_0 to p'. Note that in this way, for each pair of states $p, p' \in S$, we can produce a path $p \rightsquigarrow q_0 \rightsquigarrow q_0 \rightsquigarrow p'$ of length exactly 30. For example: Consider the pair q_3, q'_3. There is a path of length 6 from q_3 to q_0 and a path of length 3 from q_0 to q'_3. Since we want the total length of the path from q_3 to q'_3 to be exactly 30, we sandwich a closed walk of length 21 at q_0. It is easy to see that such a closed walk exists in the NFA shown.

Corollary 6.11 *For every p prime, the language $MOD_p = \{ x \mid |x|_1 \equiv 1 \bmod p \}$ admits an NC^0 proof system.*

All the proof systems for regular languages in Section 3 are obtained by applying one of Theorems 6.1, 6.3, 6.5, in conjunction with a generic closure property.

7 Conclusion

In this paper we initiated a systematic study of the power of NC^0 proof systems. We obtained a number of upper and lower bounds, some for specific languages, some more generic. The main open question that arises from our investigation is a combinatorial characterization of all languages that admit NC^0 proof systems. Our generic results from Section 6 can be seen as a first step towards

such a characterization for regular languages. We believe that further progress essentially depends on strengthening our lower bound techniques.

Agrawal's results on constant-depth isomorphisms [1] provide a possible tool to approach our main question: if we have an NC^0 isomorphism between two languages A and B, and B admits an NC^0 proof system, then so does A. The proofs for A are taken to be the proofs for B, then we simulate the proof system for B, and to the obtained word in B we apply the inverse of the reduction and enumerate an element from A.

In fact, our work seems to bear further interesting connections to recent examinations on isomorphism of complete sets for the class NP. This work was started in the nineties in a paper by Agrawal et al. [3] where it was shown that (1) every language complete for NP under AC^0 reductions is in fact already complete under (non-uniform) NC^0 reductions (this is called "gap theorem" in [3]), and (2) that all languages complete for NP under AC^0 reductions are (non-uniformly) AC^0 isomorphic (that is, the reduction is an AC^0 bijection). This was later improved to uniform AC^0 isomorphisms [1]. It follows from a result in [2] that this cannot be improved to P-uniform NC^0 isomorphisms. Using our results on proof systems, we obtain a very simple direct proof:

Proposition 7.1 *There are sets A and B that are NP complete under NC^0 reductions but not NC^0 isomorphic.*

Proof. Let A be the NP-complete set from [14] that admits an NC^0 proof system, cf. Remark 3.8. A is NP complete under AC^0 reductions, hence by the gap theorem, under NC^0 reductions.

Let B be the disjoint union of A and Exact-OR from Section 4. Then B is complete for NP under NC^0 reductions because A reduces to B in NC^0.

If now A and B are NC^0 isomorphic, then we obtain an NC^0 proof system for B and from this, an NC^0 proof system for Exact-OR, a contradiction. □

Acknowledgments.

We thank Sebastian Müller (Prague) for interesting and helpful discussions on the topic of this paper.

References

[1] M. Agrawal. The isomorphism conjecture for constant depth reductions. *Journal of Computer and System Sciences*, 77(1):3–13, 2010.

[2] M. Agrawal, E. Allender, R. Impagliazzo, T. Pitassi, and S. Rudich. Reducing the complexity of reductions. *Computational Complexity*, 10(2):117–138, 2001.

[3] M. Agrawal, E. Allender, and S. Rudich. Reductions in circuit complexity: An isomorphism theorem and a gap theorem. *J. Comput. Syst. Sci.*, 57(2):127–143, 1998.

[4] E. Allender, D. A. M. Barrington, T. Chakraborty, S. Datta, and S. Roy. Planar and grid graph reachability problems. *Theory of Computing Systems*, 45(4):675–723, 2009.

[5] B. Applebaum, Y. Ishai, and E. Kushilevitz. Cryptography in NC^0. *SIAM J. Comput.*, 36(4):845–888, 2006.

[6] B. Applebaum, Y. Ishai, and E. Kushilevitz. On pseudorandom generators with linear stretch in NC^0. *Computational Complexity*, 17(1):38–69, 2008.

[7] B. Applebaum, Y. Ishai, and E. Kushilevitz. Cryptography with constant input locality. *J. Cryptology*, 22(4):429–469, 2009.

[8] O. Beyersdorff, S. Datta, M. Mahajan, G. Scharfenberger-Fabian, K. Sreenivasaiah, M. Thomas, and H. Vollmer. Verifying proofs in constant depth. In *Proc. 36th Symposium on Mathematical Foundations of Computer Science*, volume 6907 of *Lecture Notes in Computer Science*, pages 84–95. Springer-Verlag, Berlin Heidelberg, 2011.

[9] O. Beyersdorff, J. Köbler, and S. Müller. Proof systems that take advice. *Information and Computation*, 209(3):320–332, 2011.

[10] O. Beyersdorff and S. Müller. A tight Karp-Lipton collapse result in bounded arithmetic. *ACM Transactions on Computational Logic*, 11(4), 2010.

[11] T. Chakraborty and S. Datta. One-input-face MPCVP is hard for L, but in LogDCFL. In *Proc. of 26th FST TCS Conference, LNCS vol. 4337*, pages 57–68, 2006.

[12] S. A. Cook and J. Krajíček. Consequences of the provability of NP $\subseteq$ P/poly. *The Journal of Symbolic Logic*, 72(4):1353–1371, 2007.

[13] S. A. Cook and R. A. Reckhow. The relative efficiency of propositional proof systems. *The Journal of Symbolic Logic*, 44(1):36–50, 1979.

[14] M. Cryan and P. B. Miltersen. On pseudorandom generators in NC^0. In *Proc. 26th Symposium on Mathematical Foundations of Computer Science*, pages 272–284, 2001.

[15] L. M. Goldschlager. The monotone and planar circuit value problems are logspace complete for P. *SIGACT News*, 9(2):25–29, 1977.

[16] S. Goldwasser, D. Gutfreund, A. Healy, T. Kaufman, and G. N. Rothblum. Verifying and decoding in constant depth. In *Proc. 39th ACM Symposium on Theory of Computing*, pages 440–449, 2007.

[17] J. Håstad. One-way permutations in NC^0. *Inf. Process. Lett.*, 26(3):153–155, 1987.

[18] E. A. Hirsch. Optimal acceptors and optimal proof systems. In *Proc. 7th Conference on Theory and Applications of Models of Computation*. Springer-Verlag, Berlin Heidelberg, 2010.

[19] E. A. Hirsch and D. Itsykson. On optimal heuristic randomized semidecision procedures, with application to proof complexity. In *Proc. 27th Symposium on Theoretical Aspects of Computer Science*, pages 453–464, 2010.

[20] E. Mossel, A. Shpilka, and L. Trevisan. On ε-biased generators in NC^0. *Random Struct. Algorithms*, 29(1):56–81, 2006.

[21] P. Pudlák. Quantum deduction rules. *Annals of Pure and Applied Logic*, 157(1):16–29, 2009.

[22] N. Segerlind. The complexity of propositional proofs. *Bulletin of Symbolic Logic*, 13(4):417–481, 2007.

[23] E. Viola. Extractors for circuit sources. In *FOCS*, pages 220–229, 2011.

[24] E. Viola. The complexity of distributions. *SIAM J. Comput.*, 41(1):191–218, 2012.

[25] H. Vollmer. *Introduction to Circuit Complexity – A Uniform Approach*. Texts in Theoretical Computer Science. Springer Verlag, Berlin Heidelberg, 1999.

[26] I. Wegener. *The Complexity of Boolean Functions*. Wiley-Teubner series in computer science. B. G. Teubner & John Wiley, Stuttgart, 1987.