Proceedings Paper:
Beyersdorff, O, Vollmer, H, Datta, S et al. (2011) Verifying proofs in constant depth. In: Proceedings MFCS 2011. Mathematical Foundations of Computer Science 2011, 22-26 Aug 2011, Warsaw, Poland. Springer Verlag, 84 - 95 . ISBN: 978-3-642-22992-3. ISSN: 0302-9743.

https://doi.org/10.1007/978-3-642-22993-0_11

Verifying Proofs in Constant Depth*

Olaf Beyersdorff[1], Samir Datta[2], Meena Mahajan[3],
Gido Scharfenberger-Fabian[4], Karteek Sreenivasaiah[3], Michael Thomas[1], and
Heribert Vollmer[1]

[1] Institut für Theoretische Informatik, Leibniz Universität Hannover, Germany
[2] Chennai Mathematical Institute, India
[3] Institute of Mathematical Sciences, Chennai, India
[4] Institut für Mathematik und Informatik, Ernst-Moritz-Arndt-Universität Greifswald, Germany

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 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 [12] as a polynomial-time computable function f that has as its range exactly all strings of L. In this setting, pre-images of f are considered as proofs for elements $x \in L$. Finding such a proof might be difficult, but verifying the validity of a proof can be done efficiently. In the last decades, 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. [21] 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 [20] studies quantum proof systems, Cook and Krajíček [11] introduce proof systems that may use a limited amount of non-uniformity (see also [8, 9]), and Hirsch and Itsykson [17, 18] 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.

* Research supported by a DAAD/DST grant, DFG grant VO 630/6-2, and by a grant from the John Templeton Foundation

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) 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. [15] 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 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 [5,6,13,16,19]. One of the main problems in this area is to construct pseudorandom generators which are computed by NC^0 circuits [5,6,13,19]. This question asks for NC^0-computable functions for which the range is hard to distinguish from a uniform distribution. 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. We note that Cryan and Bro Miltersen [13] 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. This implies that lower bound techniques which are used against restricted circuit classes (cf. [22,23]) are not directly applicable to show lower bounds for NC^0 proof systems. The proof techniques we use are combinatorial arguments tailored towards our specific problems.

This paper is organized as follows. We start in Sect. 2 by defining the concept of NC^0 proof systems and make some initial observations. In Sect. 3 we construct NC^0 proof systems for several languages of different type. This is followed by Sect. 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 Sect. 5 we generalize some of the ideas for NC^0 proof systems from

Sect. 3 to obtain proof systems for large classes of regular languages. Finally, we conclude in Sect. 6 with some discussion and future perspectives.

2 Definitions

A function $f : \{0,1\}^* \longrightarrow \{0,1\}$ is said to admit an NC^0 proof system if there exists a family of Boolean circuits (see, e.g., [22]) $\left(C_n\right)_{n\geq 1}$ satisfying the following conditions:

1. For all $n \geq 1$, $C_n \colon \{0,1\}^{m(n)} \to \{0,1\}^n$, where $m : \mathbb{N} \longrightarrow \mathbb{N}$.
2. For all n and for all words $x \in \{0,1\}^{m(n)}$, $C_n(x) \in f^{-1}(1)$.
3. 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 constants c, d, each C_n has size n^c, depth d, and is built using AND, OR, and NOT gates of bounded (constant) fan-in.

That is, the circuit family has as its range exactly the set $f^{-1}(1)$. (Note the non-standard size bound: we require the circuit size to be polynomial in the output length, not input length.)

A function $f : \{0,1\}^* \longrightarrow \{0,1\}$ is said to admit an AC^0 proof system if there exists a family of Boolean circuits $f^{-1}(1)$ as above, with the only difference that this time the circuits are allowed to use unbounded fan-in AND and OR gates.

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 *DLOGTIME-uniform*. (For more formal definitions, we refer the reader to [22].)

We remark that all lower bounds we will present in the sequel of this paper hold even for non-uniform proof systems, while all upper bounds will yield DLOGTIME-uniform proof systems.

For a language $L \subseteq \{0,1\}^*$, we say that L admits an NC^0 proof system 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 $y \in L$.

Since the circuit must always produce a string in L, we cannot construct such proof systems if a language has "gaps"; if for some n, $L \cap \{0,1\}^{=n} = \emptyset$, then we cannot define C_n. We therefore allow circuits that are "empty"; C_n is empty if and only if $L \cap \{0,1\}^{=n} = \emptyset$.

We observe that AC^0 proof systems do exist for every NP-set. In fact, a more general statement is true.

Proposition 1 (Folklore). *Every language in* NP *admits an* AC^0 *proof system. Every recursively enumerable language admits a constant-depth proof system.*

As mentioned already in the introduction, Cryan and Bro Miltersen [13] exhibit an NP-complete language that admits even an 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.

Proposition 2. *There are non-trivial languages in* NP *that do not admit any P-uniform* NC^0 *proof system.*

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. (Of course, this does not imply that all languages in NC^1 admit such a proof system—in particular we show in Section 4 explicit examples of particularly simple languages (such as 0^*1^*) that provably do not admit 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. *The word problem for finite groups admits an* NC^0 *proof system.*

Proof. We describe the circuit $C_n : \{0,1\}^{n-c} \to \{0,1\}^n$ (where we assume that each element of the finite group G is coded by a bit string c bits long). Since the word problem contains only words of lengths divisible by c, we produce circuits only for value of $n = cn'$ divisible by c. Given a sequence $B_1 \ldots B_{n'-1}$ as proof (input), the circuit C_n constructs the string $\langle h_1, \ldots, h_{n'}\rangle$ as follows: For $1 \leq i \leq n' - 1$, if B_i is a valid description of a group element g, then $g_i = g$, otherwise $g_i = e$. Also, $g_0 = g_{n'} = e$. Then h_i is the description of the group element $g_{i-1}^{-1} g_i$. It is easy to verify that $\prod_{i=1}^{n'} h_i = e$ and that all possible strings satisfying this property are generated. □

Corollary 4. *The parity function admits an* NC^0 *proof system.*

In proving Proposition 3 we had crucially used all the three group axioms: associativity, existence of an identity and existence of inverses. We can relax some of these axioms and still get an NC^0 proof system. As an example consider the language $L_{OR} = \{w = w_1 \ldots w_n \in \{0,1\}^* : \bigvee_{i=1}^n w_i = 1\}$. The *OR* operation is associative and has an identity, but all elements do not have an inverse. Yet, we are able to show that:

Proposition 5. *The language* L_{OR} *admits an* NC^0 *proof system.*

If we relax the group axioms to monoid axioms (i.e. drop associativity), we do not always get an NC^0 proof system for the corresponding word problem, as we show in Section 4. On the other hand, under some connectivity assumptions of the graph corresponding to the monoid, we are able to show that the word problem indeed admits an NC^0 proof system. It is an interesting open question as to what happens if we drop some other axiom such as that of existence of identity or existence of inverse or even closure under the operation.

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 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 $j, k \in \{0, \ldots, k-1\}$ 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 6. *L_{BWDR} admits an NC^0 proof system.*

Now we consider the addition of two numbers, *i.e.*, 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.

Proposition 7. *f_+ admits an NC^0 proof system.*

As a corollary, we can see that comparison admits an NC^0 proof system. Formally, define $f_\leq : \{0,1\}^n \times \{0,1\}^n \to \{0,1\}$ as follows: $f_\leq(a,b) = 1 \iff A \leq B$, where a, b are the n-bit binary representations of numbers A, B.

Corollary 8. *$f_\leq$ admits an NC^0 proof system.*

We now consider a P-complete language, Grid Circuit Value Problem. 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).

Proposition 9. *The Grid Circuit Value Problem is P-complete.*

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

Proposition 10. *The Grid Circuit Value Problem admits an NC^0 proof system.*

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

Lemma 11. *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 12. *If $A, B \subseteq \{0,1\}^*$ admit NC^0 proof systems, then so does $A \cup B$.*

Lemma 13. *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 an NC^0 proof system.*

Remark 14. Note that in proving Lemma 12, 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 certain cases, the complement of a language with an NC^0 proof system also has an NC^0 proof system. For example:

Lemma 15. *Let $L \subseteq \{0,1\}^*$ have the property that there is a constant c such that for each n, $|L \cap \{0,1\}^n| \geq 2^n - k$. That is, at each length, at most c strings of that length are not in L. Then L admits an NC^0 proof system.*

4 Lower bounds

We now consider languages in PTime, some of them even regular, which do not admit NC^0 proof systems. 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 be some counting feature.

4.1 Lower bounds on depth

We begin with our main concrete example of a non-NC^0-enumerable language. To show that it is not enumerable, we derive in Theorem 16 a criterion which implies non-constant lower bounds for the depth of an enumerating circuit family. All further examples will be in some way reduced to this one.

Exact Counting. Consider the function Exact-Count$_k^n$ on n bits: it evaluates to 1 if and only if exactly k of the input bits are 1.

For each length n there are exactly $\binom{n}{k}$ words in Exact-Count$_k^n$. 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$_k^n$ the parameters $t(n)$ and $\ell(n)$ defined in the theorem below take the values $\binom{n}{k}$ and k, respectively. It might be good to keep this example in mind, when reading the following theorem and its proof (in the appendix).

Theorem 16. *Let L be a language and $\ell, t : \mathbb{N} \to \mathbb{N}$ functions such that for each length n there are $t(n)$ distinct settings to subsets of $\ell(n)$ bits $x_{i_1}, \ldots, x_{i_{\ell(n)}}$ such that each of these partial configurations enforces a fixed value to each of the remaining bits. Then the depth of each circuit family that enumerates L is at least* $\log\log t(n) - \log \ell(n)$.

For our main example Exact-Count$_k^n$ the parameters ℓ and t of the theorem evaluate to $\ell(n) = k$ and $t(n) = \binom{n}{k}$, 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, so in this case Exact-Count$_k^n$ does not admit an NC^0 proof system. Note that for a constant k this language is even regular.

We continue and give further examples of languages without NC^0 proof system and state an open problem.

$\neg Th_{k+1}^n$ and dually Th_{n-k}^n for sub-linear k. Let Th$_a^n$ 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$_k^n$. So they also yield the same set of parameters.

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$_1^{n+1}$ via the NC^0 mapping $w_1 \ldots w_n \longmapsto x_1 \ldots x_{n+1}$, 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$_1^{n+1}$, 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.

Open problem: Majority. Majority is the language which consists of those words which have at least as many 1's as there are 0's.

Does Majority admit an NC^0 proof system? The lower bound on the depth of enumerating circuits given by Theorem 16 is only 2 (using an approximation for Catalan's number). A majority of the authors believes that Majority does not admit an NC^0 proof system. But this seems hard to prove.

4.2 List enumerations

Consider a circuit $C : \{0,1\}^m \longrightarrow \{0,1\}^{tn}$. 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 an 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$_k^n$ 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$_1^n$ enumerated by a single circuit is small, and hence Exact-Count$_1^n$ **requires** $\Omega(n)$ lists. We will use this in Theorem 19 to prove a lower bound for the list length of the language of all permutation matrices.

Lemma 17. *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 16, replacing $t(n)$ by $|L^{=n}|$. □

Remark 18. 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.

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 17 gives the following:

Theorem 19. *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.*

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.

5 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.

Theorem 20. *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. (Sketch.) The idea is to give a word x and the states of the NFA on some accepting run of M on x. Instead of giving the entire state sequence, only the states after every k steps are given. The NC^0 circuit checks, for each "block" of x, if the states before and after the block are consistent with x. If so, this part of x is output as is; otherwise, it is replaced by a string of length $\leq k$ that takes M from the state at the beginning of the block to an absorbing final state. □

Observe that the OR and the Exact-OR are both star-free languages but the complementations 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 21. *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 22. *Let r be a strict star-free expression describing a language $L = L(r)$. Then L admits an NC^0 proof system.*

Proof. (Sketch) Strict star-free expressions can be written as sums of star-free sum-free terms. For each such term, Theorem 20 can be used, and finally, Lemma 12 puts them together. □

Theorem 20 essentially characterizes functions like OR. On the other hand, the parity function, that has a 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 23. *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. (Sketch.) 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 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 the total 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 thus need 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:

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

Let g be the greatest common divisor of all the numbers in L. Though L is infinite, it has a finite subset L' whose gcd is g. Choose a set $S \subseteq Q$ 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. 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 $\equiv r_p \pmod g$.

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

Claim. 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.

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. Putting these facts together, we can construct a proof that can be corrected in NC^0. □

Corollary 24. *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 we have seen in Section 3 for regular languages are obtained by applying one of Theorems 20, 22, 23, possibly in conjunction with a generic closure property.

6 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 Sect. 5 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. In particular, we ask whether Majority admits an NC^0 proof system.

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

Proof. Let A be the already mentioned NP-complete set from [13] that admits an NC^0 proof system. A is NP-complete under AC^0-reductions, hence by the gap theorem, under NC^0-reductions.

Let K be any set in NP but not in $\mathrm{NTIME}(n)$; such a language exists by the nondeterministic time hierarchy theorem. Let B be the disjoint union of A and K. Then B is complete for NP under NC^0 reductions because A reduces to B in NC^0. And B is not in $\mathrm{NTIME}(n)$ because K reduces to B in $\mathrm{DTIME}(n)$.

If now A and B are NC^0-isomorphic, then we obtain an NC^0 proof system for B. This implies that B is in $\mathrm{NTIME}(n)$, a contradiction. □

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.

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, J. Köbler, and S. Müller. Proof systems that take advice. *Information and Computation*, 209(3):320–332, 2011.
9. O. Beyersdorff and S. Müller. A tight Karp-Lipton collapse result in bounded arithmetic. *ACM Transactions on Computational Logic*, 11(4), 2010.
10. 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.
11. 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.
12. S. A. Cook and R. A. Reckhow. The relative efficiency of propositional proof systems. *The Journal of Symbolic Logic*, 44(1):36–50, 1979.
13. 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.
14. L. M. Goldschlager. The monotone and planar circuit value problems are logspace complete for P. *SIGACT News*, 9(2):25–29, 1977.
15. 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.
16. J. Håstad. One-way permutations in NC^0. *Inf. Process. Lett.*, 26(3):153–155, 1987.
17. 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.
18. 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.
19. E. Mossel, A. Shpilka, and L. Trevisan. On ε-biased generators in NC^0. *Random Struct. Algorithms*, 29(1):56–81, 2006.
20. P. Pudlák. Quantum deduction rules. *Annals of Pure and Applied Logic*, 157(1):16–29, 2009.
21. N. Segerlind. The complexity of propositional proofs. *Bulletin of Symbolic Logic*, 13(4):417–481, 2007.
22. H. Vollmer. *Introduction to Circuit Complexity – A Uniform Approach*. Texts in Theoretical Computer Science. Springer Verlag, Berlin Heidelberg, 1999.
23. I. Wegener. *The Complexity of Boolean Functions*. Wiley-Teubner series in computer science. B. G. Teubner & John Wiley, Stuttgart, 1987.

Appendix

The appendix contains all proofs which are omitted in the main part due to space restrictions.

Proposition 1. *Every language in* NP *admits an* AC^0 *proof system.*
Every recursively enumerable language admits a constant-depth proof system.

Proof. (Sketch.) Let L be accepted by the nondeterministic Turing machine M. The proof of a word $x \in L$ is an encoding of an accepting sequence of configurations of M on input x. 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. This can be done in constant depth, and if the run-time of M is polynomial, then the checking circuit is of size polynomial in the output word. □

Proposition 2. *There are non-trivial languages in* NP *that do not admit any P-uniform* NC^0 *proof system.*

Proof. If a language L has a P-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). □

Proposition 5. *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 $a = a_1 \dots a_n$ and $b = b_1 \dots b_{n-1}$, and outputs a sequence $w = w_1 \dots w_n$ where

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

Here for notational convenience we assume that $b_0 = 0, b_n = 1$. Notice that if b_i's indeed code the prefix OR's of a_j's at all positions 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 in w_i. Thus C_n is an onto map from $\{0,1\}^{2n-1} \to L_{OR} \cap \{0,1\}^n$ completing the proof. □

Proposition 6. 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 $i \in \{0, \ldots, L-1\}$, 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. □

Proposition 7. *f_+ admits an NC^0 proof system.*

Proof. The circuit $C_n : \{0,1\}^{3n}$ maps three strings $\alpha = \alpha_{n-1}, \ldots, \alpha_0$, $\beta = \beta_{n-1} \ldots \beta_0$ and $\gamma = \gamma_n \ldots \gamma_1$ (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 $\mathrm{Th}_2^3(\alpha_i, \beta_i, \gamma_i) = \gamma_{i+1}$ then $a_i = \alpha_i, b_i = \beta_i$ otherwise, set a_i, b_i arbitrarily under the constraint that $\mathrm{Th}_2^3(a_i, b_i, \gamma_i) = \gamma_{i+1}$. Also set $s_i = a_i \oplus b_i \oplus \gamma_i$. Set $s_n = \gamma_n$. □

Corollary 8. *$f_\leq$ 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 $a < b$. □

Proposition 9. *The Grid Circuit Value Problem is P-complete.*

Proof. It is easy to see that this problem is contained in P. To see that it is P-hard, we reduce Circuit Value Problem to it under, say, $\mathsf{DLogspace}$ reductions. First make the circuit planar by using the usual cross-over gadget [14] to remove all crossings. Now, embed the circuit in the grid by using a method similar to the one used in [4, 10] 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. □

Proposition 10. *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 setting one entry of each truth table to make it consistent with the values in v. x describes the values (from v) corresponding to inputs and b the value of the output gate. □

Lemma 12. *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. □

Lemma 15. *Let $L \subseteq \{0,1\}^*$ have the property that there is a constant c such that for each n, $|L \cap \{0,1\}^n| \geq 2^n - k$. That is, at each length, at most c strings of that length are not in L. Then L admits an 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 output 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. □

Theorem 16. *Let L be a language and $\ell, t : \mathbb{N} \to \mathbb{N}$ functions such that for each length n there are $t(n)$ distinct settings to subsets of $\ell(n)$ bits $x_{i_1}, \ldots, x_{i_{\ell(n)}}$ such that each of these partial configurations enforces a fixed value to each of the remaining bits. Then the depth of a circuit family that enumerates L is at least* $\log\log t(n) - \log \ell(n)$.

Proof. Let $f : m(n) \longrightarrow n$ 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 $\ell(n)2^{d(n)}$ bits of the proof. Thus there is a setting to $r(n) = \ell(n)2^{d(n)}$ 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, the sum over the $|f^{-1}(w_j)|$ and therefore also $t(n)2^{m(n)-r(n)}$ is bounded from above by $2^{m(n)}$. This immediately gives the estimate

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

□

Theorem 19. *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-Count$_1^n$. By Lemma 17, each C_i can construct at most N^n matrices where $N = 2^{2^d}$ (it has at most N choices for each row). But the total number of choices must be at least the number of permutation matrices. Thus $n! \leq tN^n$ and if $d \in O(1)$, t must be exponentially large. □

Theorem 20. *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. The hypothesis is that from each "useful" 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 $\text{fin}(q)$ (i.e., $\text{fin}(q)$ "finalizes" q). Clearly, $\delta(q, \text{fin}(q)) \in F'$.

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:

$$w^0 = x^0$$

$$w^1 = \begin{cases} x^1 & \text{if } \ q^2 \in \delta(q^0, x^0 x^1) \\ \text{fin}(\delta(q_0, x^0)) & \text{otherwise.} \end{cases}$$

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

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. □

Theorem 22. *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 12).

The following claim shows that this is indeed true:

Claim. 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 20 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 23. *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 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:

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

Let g be the greatest common divisor of all the numbers in L. Note that though L is infinite, it has a finite subset L' 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. 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. 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 L. □

Claim. 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 26 below. □

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 5 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 5, such a word exists.

□

Lemma 26 (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 $j-1$. 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. $\square$