Languages in AC^1 defined by iterating finite transducers

Robert Surówka, Kenneth W. Regan

Abstract

We present an explicit construction of AC^1 circuits that simulate finite state transducer iteration. Our approach yields circuits with only $\Theta(n \log(n))$ linear fan-in gates, whereas the circuits built from standard proof of $\mathsf{L} \subseteq \mathsf{AC}^1$ use matrix multiplication and hence have $\Theta(n^2 \log(n))$ such gates.

1 Introduction

A Finite State Transducer (FST) is a finite automaton that while processing its input, sequentially writes an output. Several kinds of FST's have been studied in the literature. The earliest and best known are the Moore machine and the Mealy machine (cf. [Hopcroft, Ullman 1969]), which differ mainly in whether an output character is printed upon entering a state or while processing a transition. An extension of the latter called a *generalized sequential machine* (also cf. [Hopcroft, Ullman 1969]) allows outputting the empty string λ on some transitions, rather than a character. Further extensions allow more than one output character, and/or a final character printed in a halting state, and go by names such as "sub-sequential machines." We revert to the generic name for our formal definition:

Definition 1. A *finite-state transducer* is an 8-tuple $M = (Q, \Sigma, \Gamma, \delta, \rho, s, F, \phi)$ where $F \subseteq Q$ is the set of good final states, $s \in Q$ is the start state, Σ and Γ are the input and output alphabets, $\delta : Q \times \Sigma \to Q$ is the transition function, $\rho : Q \times \Sigma \to \Gamma^*$ is the output function, and $\phi : F \to \Gamma^*$ is the final-output function. An FST M computes a partial function $f : \Sigma^* \to \Gamma^*$ if for all $x \in \Sigma^*$, the computation $M(x)$ ends in a state $r \in F$ if and only if x is in the domain of f, whereupon the outputs of ρ concatenated with the final output $\phi(r)$ equal the value $f(x)$.

For some examples, the function $f(x) = xb$ where b is a parity-check bit for $x \in \{0,1\}^*$ is computed by the FST with two states $Q = F = \{s, o\}$, transitions $\delta(s,0) = s$, $\delta(o,0) = o$, $\delta(s,1) = o$, $\delta(o,1) = s$; outputs $\rho(q,b) = b$, and final outputs $\phi(s) = 0$, $\phi(o) = 1$. Note that $\phi(r)$ could alternately be regarded as the output on a final end-of-string marker. The one-state FST with $\delta(s,c) = cc$, $\phi(s) = \lambda$ doubles each character c in the input string, and every other *string homomorphism* is similarly computed.

Every FST-computable function f can be computed by $O(\log n)$-depth, $O(n)$-size Boolean circuits (under a homomorphic binary encoding of Σ and Γ, which we presume from here onward), by the general technique of "parallel prefix sum." These circuits also have bounded fan-in, so they comply with the definition of NC^1 circuits, while unbounded fan-in, log-depth, polynomial-size circuits are said to define the class AC^1 (for both uniform circuit classes, see [Cook 1985]). The classes of deterministic and nondeterministic logarithmic space are sandwiched between them, that is

$$\mathsf{NC}^1 \subseteq \mathsf{L} \subseteq \mathsf{NL} \subseteq \mathsf{AC}^1,$$

but none of these inclusions is known to be proper, nor is NC^1 even known to differ from the class NP.

To probe the relationships between these classes further—especially when we insist on linear or nearly-linear rather than polynomial bounds on the sizes of the circuits—we consider problems defined by *iterating* FSTs. Iterating an FST means giving its output from the previous iteration as the input to the next one. An example of a nontrvial problem that can be solved by iterating an FST is whether a given string of parentheses correctly balances. The FST M for this problem checks whether the first character is '(', rejecting if not, and then groups input characters in pairs, also rejecting unless the last pair is followed by a final character ')'. For each pair it uses two transitions to effect the outputs

$$((\to (,\quad)) \to),\quad)(\to \lambda,\quad () \to \lambda.$$

Then x is balanced if and only if $M(x)$ accepts and is balanced, while $|M(x)| < |x|/2$. Hence the brackets in the input string match if and only if the sequence of iterations ends with empty output, and there are at most $\log_2 |x|$ such iterations.

When an FST's output is always either shorter than its input or smaller than a certain constant, then such iteration is guaranteed either to converge to the empty string or to reject (infinitely cycle over a finite number of strings). We pose the problem of deciding whether a given string is (at some point) produced as an output of such an iteration of a given FST on a given input. This problem is in L, because we can simulate the $\log_2 n$ iterations in parallel working left-to-right on the strings, hence it is in AC^1. However, we note:

- The circuits built from the standard proof of $\mathsf{L} \subseteq \mathsf{AC}^1$ use matrix multiplication, and hence have $\Omega(n^2 \log(n))$ gates of linear fan-in.
- Composing the NC^1 circuits for each iteration gives $O(n \log n)$ size, but $\Theta(\log^2 n)$ depth, which is not valid for AC^1.

Hence there is interest in building explicit AC^1 circuits for this problem that use only $O(n \log(n))$ gates of linear fan-in. Our main theorem does so.

After defining *fixed-stride* FSTs and formalizing the iteration problem, we start with three lemmas about relationships between functions and AC^1 circuits. We follow with a theorem for the basic case of an FST with fixed stride 2, in which our idea shows most clearly. Subsequently we build upon it to present a theorem and corollaries for broader and broader cases, finally obtaining the result for FSTs without a fixed stride (yet with another, weaker, restriction), and for the general question of whether a given word is produced at any level of the iteration.

2 Definitions

Definition 2. An FST M has *a fixed stride* k to t, if $\forall_i$ M outputs t characters as a result of reading characters from ki to $k(i+1)-1$ of the input word. For $t = 1$, we just say that M has *a fixed stride* k.

Definition 3. For any FST M we define $L^*(M) = \{x : M^*(x) = \lambda\}$, where λ is the empty string. $L^*(M)$ is a language of multiple passes of M. Additionally we define for any $w \in \Gamma^*$ a $L^*_w(M) = \{x : \exists_{n\in\mathbb{N}}\ M^n(x) = w\}$. Finally note that $L^*(M) \equiv L^*_\lambda(M)$.

3 Results

Lemma 1. *A function $f : D \to V$ such that $|D| = n^{O(1)}$ and $|V| = n^{O(1)}$ can be computed by a circuit having constant depth, polynomial size in n and unbounded fan-in.*

Proof. Let us assume that k bits/wires are sufficient to encode any $d \in D$, and m bits/wires are sufficient to encode any $v \in V$. We represent the function f by $n^{O(1)}$ pairs of sheaves of wires. Each pair has an "input" sheaf consisting of k wires, and an "output" sheaf having m wires, which map any possible argument to f to value f returns for it. As an input to f we supply k wires coding an element x of D for which we want to obtain a value y from V. Computing f consists of the following steps:

1. NXOR x with each input sheaf of each pair. Those are bounded fan-in pairwise NXOR-s, i.e. $\forall i \in [1, k]$ we NXOR i-th bit of x with i-th bit of each input sheaf. As a result we obtain $|D|$ k-wire sheaves, where exactly one of them contains only ones.

2. We perform unbounded fan-in AND-s on each of those sheaves, now from each input sheaf we will have a single wire, and exactly one of them will be 1—the one whose input sheaf equaled x.

3. Now we perform pairwise ANDs between the single wires from previous step and each wire of their respective "output" sheaves. In the end, there will be exactly one 'output' sheaf (having "input" sheaf equaling x) whose bits will be unchanged, whereas all other output sheaves will have their bits set to 0.

4. Let us perform now for all $i \in [1, d]$ unbounded fan-in OR-s, each OR taking as input all i-th wires from all "output" sheaves. The result of this step will be a single m-wire sheaf equal to the "output" sheaf from previous step, which means it will be equal to the $f(x)$.

□

Lemma 2. *A function $f : D \to V$ such that $|D| = n^{O(1)}$ and $|V| = n^{O(1)}$ can be computed simultaneously for $t = n^{O(1)}$ inputs (and produce t outputs) by a circuit having constant depth, polynomial size in n, and unbounded fan-in.*

Proof. For each input we perform in parallel and in isolation the same procedure as we presented in proof of Lemma 1. Because our blow-up in size is only quadratic, the size of the circuit will be still polynomial in n. □

Lemma 3. *Assume we have two functions $f : D \to V$ and $g : V \to Y$ such that $|D| = n^{O(1)}$, $|V| = n^{O(1)}$ and $|Y| = n^{O(1)}$, represented as wires mapping all possible "inputs" of those functions to their respective "outputs". We can compute a function $h = g \circ f$ (which will be also represented as pairs of groups of wires containing respective "inputs" and "outputs") in constant depth, polynomial size in n, and unbounded fan-in.*

Proof. The function h will have pairs of groups of wires such that each "input" group will represent an entity from D and each "output" group will represent an entity from Y. To produce the wire representation of h from wire representations of f and g, we compute g on each "output" group of wires from f. By lemma 2 we can do this in constant depth, polynomial size in n and with unbounded fan-in. Afterwards we route (no gates needed) wires representing "outputs" of g to be next to the respective "inputs" of f ("outputs" of which where the "inputs" to h). In this way we obtain the required representation of h. □

Theorem 1. *Let M be an FST with fixed stride 2. Then $L^*(M) \in AC^1$.*

Proof. We can assume that the initial input word x has length that is a power of 2. We can do that, since we can always pad it with a special additional character ω, to the closest power of 2. Additionally, we assume that the FST M on each of its edges reads 2 characters and outputs a character, since any FST with fixed stride 2 can be rewritten in such a way (some of those edges may read up to 2 ω-s). Let us divide x into $|x|/2$ pairs of consecutive characters. During the first pass of the FST M, each of those pairs will be translated into a single character. We can define a function, that given a starting state in which FST M is upon starting to read a pair, and that pair itself, returns the state in which FST M is after reading that pair, and character outputted by the FST M in the process. Let us define such a function as $g_1 : Q \times \Gamma^2 \to Q \times \Gamma$. Let us notice that we can compute that function for $|x|^{O(1)}$ arguments in constant depth (due to Lemma 2). We should note that in our circuit we represent all functions in a way allowing Lemmas 1 to 3 to be applied to them.

Let us denote the characters of the i-th pair as $c_{1,i}$ (1 meaning, that this pair is an input to the first pass). Let us define $f_{1,i} : Q \to Q \times \Gamma$ such that $\forall q \in Q : f_{1,i}(q) = g_1(q, c_{1,i})$. We can build such an $f_{1,i}$ by computing g_1 for all $(q \in Q, c_{1,i})$ and match outputs with their respective inputs. Let us note that due to $i \in [0, \frac{|x|}{2} - 1]$ and Lemma 2, we can build all $f_{1,i}$ functions in an AC^0 sub-circuit.

Before moving forward let us introduce two new notations. For any function h with codomain of the form $Q^a \times \Gamma^b$, for any a and b, for any possible argument to h arg let $h^Q(\text{arg})$ be just the first a elements of $h(\text{arg})$ (i.e. the states), and let $h^C(\text{arg})$ be just the last b elements of $h(\text{arg})$ (i.e. the characters).

Functions $f_{1,i}$ are what our circuit prepares as its "output" from the first pass of the FST M. For any i-th pair of the first pass a function $f_{1,i} : Q \to Q \times \Gamma$ states for each state the FST M may be when starting to read that pair, in which state the FST M is after finishing to reading it, and which character is outputted. To simulate the second pass of FST M, first our circuit will compute functions $g_{2,i} : Q \to Q \times \Gamma^2$ (for $i \in [0, \frac{|x|}{2^2} - 1]$) such that $g_{2,i}(q) = (f^Q_{1,2i+1}(f^Q_{1,2i}(q)),\ f^C_{1,2i}(q),\ f^C_{1,2i+1}(f^Q_{1,2i}(q)))$. Any $g_{2,i}$, given a state in which the FST M could be upon starting to read the $2i$-th pair of the first pass, says in which state FST M would end reading the $2i{+}1$-th pair (during the first pass), and which 2 characters would be outputted in the process. Due to Lemma 3 we can build $g_{2,i}$ in constant depth, and polynomial size.

Let us define now $f_{2,i} : Q^2 \to Q^2 \times \Gamma$ (for $i \in [0, \frac{|x|}{2^2} - 1]$) as: $f_{2,i}(q_{a_1},\ q_{a_2}) = (g^Q_{2,i}(q_{a_1}),\ g_1(q_{a_2},\ g^C_{2,i}(q_{a_1})))$. Each $f_{2,i}$ tells us, given a state in which FST M could be when starting to read pair $2i$ in first pass, and a state in which the FST M could be in when starting to read pair i in the second pass, what would be the states in which the FST M finishes reading pair $2i{+}1$ in the first pass and pair i in the second pass—and which character (during the second pass) is outputted in the process. Once again, we can build all $f_{2,i}$ in constant depth—we just need to compute their "inside" functions for all possible inputs, and then properly pair outputs with inputs. The functions $f_{2,i}$ are the "output" of our circuit from the simulation of the second pass of the FST M.

We described the operations our circuit performs for the first and the second pass of the FST M. It can continue to perform the same operations for subsequent passes. As an input to any given pass $p+1$ it will have $\frac{|x|}{2^p}$ functions of the form $f_{p,i} : Q^p \to Q^p \times \Gamma$, one for each of the pairs in the word the FST M reads on its p-th pass. Each $f_{p,i}$ takes p states as arguments, where the j-th of them (for $j \in [1, p-1]$) is a state in which the FST M could be in its j-th pass, when starting to read the first pair from those that generated the i-th pair of the input to pass p. The p-th argument is the state in which the FST M may be when starting to read the i-th pair of pass p itself. As an output, $f_{p,i}$ tells us, that providing the "starting" states we inserted, what would be the "ending states" of the FST M after reading each sequence of pairs responsible for i-th pair of pass p, the "ending state" of reading the i-th pair of pass p itself, as well as the character generated by that pair.

To produce output of any given pass $p+1$, first we compute functions $g_{p+1,i} : Q^p \to Q^p \times \Gamma^2$ (for $i \in [0, \frac{|x|}{2^{p+1}} - 1]$) as:

$$g_{p+1,i}(\vec{q}) = (f^Q_{p,2i+1}(f^Q_{p,2i}(\vec{q})),\ f^C_{p,2i}(\vec{q}), f^C_{p,2i+1}(\vec{q}))) \tag{1}$$

(the functions $f_{p,i}$ are the input to the p-th pass), where $\vec{q} = q_{a_1}, q_{a_2}, \ldots, q_{a_p}$. Let us notice that p is bounded by $lg|x|$, which means that sizes of domain and codomain of functions $f_{p,2i}$ and $f_{p,2i}$ are polynomial. Therefore, due to Lemma 3, we can build all $g_{p+1,i}$ in an AC^0 sub-circuit.

Next we produce functions $f_{p+1,i} : Q^{p+1} \to Q^{p+1} \times \Gamma$ (for $i \in [0, \frac{|x|}{2^{p+1}} - 1]$) as:

$$f_{p+1,i}(\vec{q},\ q_{a_{p+1}}) = (g^Q_{p+1,i}(\vec{q}),\ g_1(q_{a_{p+1}},\ g^C_{p+1,i}(\vec{q}))), \tag{2}$$

which constitute the output of the $p+1$-th pass. Because of the same reasons as with $g_{p+1,i}$ functions, we can also produce all $f_{p+1,i}$ by an AC^0 sub-circuit. Therefore, from the input to any pass $p+1$ we can produce the output of that pass in constant depth, unbounded fan-in and polynomial size in $|x|$.

We should also note that the size of any $f_{p,i} : Q^p \to Q^p \times \Gamma$ is up to polynomial in $|x|$. This is because the FST M needs exactly $lg|x|$ (lg being logarithm with base 2) passes to reach an output of single character. The one with the "biggest" size is $f_{lg|x|,i}$. It can accept $|Q|^{lg|x|}$ different argument sequences, and since $|Q|$ is constant, therefore the number of possible inputs to $f_{lg|x_0|,i}$ is $|x|^{\Theta(1)}$. Additionally, for each possible input, we need to remember an output for it, which is just a sequence of length in $\Theta(lg|x|)$. Therefore the total size of $f_{p,i}$ is $|Q|^{lg|x|}(2lg|x|Q_b + \Gamma_b)$, where $Q_b = \lceil lg|Q| \rceil$ and $\Gamma_b = \lceil lg|\Gamma| \rceil$ are numbers of bits needed to respectively represent a state and a character of M. Furthermore, we should note that the total number of all $f_{p,i}$ for all p and i, is bounded by $|x|$. Therefore the size needed to remember all $f_{p,i}$ is polynomial in $|x|$.

At this point we see that we will produce functions $f_{p,i}$ for all of the $lg|x|$ passes. Furthermore the total size of all those functions is polynomial in $|x|$. We also note that at the very end we will just have a single function $f_{lg|x|,0} : Q^{lg|x|} \to Q^{lg|x|} \times \Gamma$, and $f^C_{lg|x|,0}(q_1, q_1, \ldots, q_1)$ is the final single output character. This is because we know that the FST M started each pass in its starting state q_1. Due to

Lemma 1 we can obtain the output of $f_{lg|x|,0}$ in an AC^0 circuit. M, upon obtaining that single character, will still perform subsequent passes on it, but now any output will have length up to 1. Therefore M will enter a cycle after up to $|\Gamma|$ steps or will output a λ at some point. Our circuit can simulate M for that constant number of steps in AC^1. If λ is an output during that simulation we accept, otherwise it means M would cycle infinitely, therefore we reject.

To sum up, we showed that we can build functions $f_{p,i}$ related to each pass p (up to $lg|x|$) of FST M in a single AC^0 sub-circuit per pass. Because there are $lg|x|$ of those passes, it means that we can build all those functions with use of an AC^1 circuit. Moreover, having the last function $f_{lg|x|,0}$, we obtain (in AC^0) the output character of all those passes of the FST M. Then, in AC^1, we check if M cycles or arrives at λ. □

Theorem 2. *Let M be an FST with fixed stride 2. Then $\forall_{w\in\Gamma^*} L_w^*(M) \in AC^1$.*

Proof. The proof of this theorem requires a slight augmentation to functions presented in the proof of Theorem 1; basically we will make the functions remember previous words. Now we will have $f_{p,i} : Q^p \to Q^p \times \Gamma \times \Gamma^2 \times \ldots \times \Gamma^{2^{p-1}}$. Let us note that $f_{1,i}$ has a form exactly as in the previous proof. Now any $f_{p,i}$ will not only return the "topmost" character, but also all the character sequences that are its ancestors (not including the part of the input ford). Similarly, now we will have $g_{p+1,i} : Q^p \to Q^p \times \Gamma^2 \times \Gamma^4 \times \ldots \times \Gamma^{2^p}$.

Before we proceed, let us introduce a new notation: For any $f_{p,i}$, any argument arg, and $s \in [0, p-1]$ let $f^s_{p,i}(\text{arg})$ be the Γ^{2^s} part of the $f_{p,i}(\text{arg})$. We do similar for $g_{p+1,i}$ and $s \in [1, p]$.

Having functions $f_{p,i} : Q^p \to Q^p \times \Gamma^{2^p-1}$ as input to $p+1$-th pass, we now produce $g_{p+1,i} : Q^p \to Q^p \times \Gamma^{2^{p+1}-2}$ as:

$$\begin{aligned} g_{p+1,i}(\vec{q}) =&(f^Q_{p,2i+1}(f^Q_{p,2i}(\vec{q})),\ f^0_{p,2i}(\vec{q}), f^0_{p,2i+1}(f^Q_{p,2i}(\vec{q})),\\ &f^1_{p,2i}(\vec{q}), f^1_{p,2i+1}(f^Q_{p,2i}(\vec{q})), \ldots,\ f^{p-1}_{p,2i}(\vec{q}), f^{p-1}_{p,2i+1}(f^Q_{p,2i}(\vec{q}))), \end{aligned} \tag{3}$$

and afterwards we obtain $f_{p+1,i} : Q^{p+1} \to Q^{p+1} \times \Gamma^{2^{p+1}-1}$ by:

$$f_{p+1,i}(\vec{q},\ q_{a_{p+1}}) =(g^Q_{p+1,i}(\vec{q}),\ g_1(q_{a_{p+1}},\ g^1_{p+1,i}(\vec{q})), g^1_{p+1,i}(\vec{q}), \ldots, g^p_{p+1,i}(\vec{q})). \tag{4}$$

An analogous argument as for Theorem 1 can be made, that all the functions have polynomial size. At the end we have $f_{lg|x|,i}$, and $f_{lg|x|,i}(q_1, q_1, \ldots, q_1)$ as an output gives words obtained after each pass up to $lg|x|$. Afterwards, we simulate M for additionalal $|\Gamma|$ passes and remember the outputs. Then we just need to compare w to each of the obtained outputs (when comparing we ignore any trailing padding). □

Corollary 1. *Let M be an FST with fixed stride $2k$ to k. Then $\forall_{w\in\Gamma^*} L_w^*(M) \in \mathsf{AC}^1$*

Proof. We can "translate" the alphabet Γ of M, so that any sequence of k characters will be just a single character. Then we can directly apply Theorem 2, yet we need afterwards to translate all outputs back to the original alphabet before comparing with w. □

Corollary 2. *Let M be an FST with fixed stride k to t, where $k > t$. Then* $\forall_{w\in\Gamma^*} L^*_w(M) \in \mathsf{AC}^1$

Proof. Let us define an FST M_r that simulates FST M, in a way that it performs r passes of M in one pass. M_r has a fixed stride k^r to t^r. Let us take $r = \left\lceil log_2^{-1}\left(\frac{k}{t}\right)\right\rceil$ and notice that then $k^r \geq 2t^r$. M_r from reading $2k^r$ characters outputs $2t^r$ of them, and we can make M_r always pad that output to obtain a sequence of length k^r. Therefore M_r can be built to have a fixed stride $2k^r$ to k^r, which allows use of Corollary 1. However, there are two issues that must be addressed. Firstly, M_r produces intermediary words for only every r-th pass of M. To check, whether M at some pass produces w, first let us simulate $r-1$ passes of M on w and remember the outputs—we can do this in AC^1, since r is constant. Then, we check if w or any of those $r-1$ outputs equals any of the outputs of M_r. If that is not the case, we reject. Otherwise, if o_p is the output of the p-th pass of M_r that was found to match, we simulate $r-1$ passes of M on o_{p-1}. If o_{p-1} or any of the just produced outputs equals w then we accept, otherwise we reject. Secondly, since we pad any part of output of length $2t^r$ to have length k^r, now the padding is not only at the ends of the outputs, but is mixed with the original characters. Therefore, before doing any comparisons we need to filter the padding to the right of each output. This can be done in NC^1, for each output in parallel. Afterwards we can easily compare any outputs and w between each other. □

Corollary 3. *Let M be an FST for which $\exists_k \forall_{w,x\in\Gamma^*} : |x| \leq k \Rightarrow |M(wx)| < |M(w)| + k$. Then $\forall_{w\in\Gamma^*} L^*_w(M) \in \mathsf{AC}^1$.*

Proof. We can assume that M after producing an output from any k-sequence, pads that output with a special additional character ω so that it has a length of $k-1$. Additionally M upon reading any ω neither changes its current state nor outputs anything. Therefore, we can take that M has a fixed stride of k to $k-1$, which allows us to directly apply Corollary 2. □

4 Directions for future work

Even though the relation between P and NP is the most famous problem in the theoretical computer science, no one has proven a lower bound even for classes believed to be much smaller than P. A pivotal small class is ACC^0, which contains problems solvable by polynomial-size constant-depth circuits of gates that can count solutions modulo some fixed integer m. When m is prime, lower bounds have been established, but for a composite m almost nothing is known (see for instance [Chattopadhyay, Wigderson 2009]). $\mathsf{ACC}^0[6]$ is not known to be separated from NP even when the circuits are uniform. It was considered a major breakthrough when three years ago Ryan Williams [Williams 2011] separated nonuniform ACC^0 from nondeterministic exponential time.

Besides the ACC^0 frontier for polynomial-sized circuits described above, there is also a frontier posed by L. Valiant at what we might call "Linear-size NC^1". That is, Valiant [Valiant 1977] raised the still open problem of proving that SAT — or any other reasonable explicit language — does not have linear size circuits of logarithmic depth. One source of uniform quasi-linear size log-depth circuits comes from iterating an FST as we have shown. If one could additionally show that iterating FSTs may simulate such circuits, then finding limits on power of just FST iteration would be sufficient to try solve the question posed by Valiant.

There are also challenging open questions in the range NC^1, L, NL, AC^1 [Kintali 2010]. Prompted by Valiants problem it would be interesting to develop a theory of linear or quasi-linear space or circuit size for these classes. Finite state transducers play an important role in linear-time computation and their algebraic properties are already known to be extensive [Hansen 2012].

5 References

Chattopadhyay A., Wigderson A. 2009. Linear systems over composite moduli. Foundations of Computer Science, 50th Annual IEEE Symposium on. IEEE 43-52.
Cook S. 1985. A Taxonomy of Problems with Fast Parallel Algorithms. Information and Control 64: 2-22.
Hansen K. 2012. An Exposition of the Barrington-Thérien Classification. Manuscript available at `https://services.brics.dk/java/courseadmin/CT12/documents/getDocument/BarringtonTherienExposition.pdf?d=80382`
Hopcroft J., Ullman J. 1969. Formal Languages and their Relation to Automata. Addison- Wesley, Boston, USA.
Kintali S. 2010. Realizable Paths and the NL vs L Problem. arXiv preprint arXiv:1011.3840.
Valiant L. 1977. Graph-theoretic arguments in low-level complexity. In Proceedings of the 2nd Annual Conference on Mathematical Foundations of Computer Science, Springer LNCS 53: 162-176.
Williams R. 2011. Non-uniform ACC circuit lower bounds, Computational Complexity (CCC), IEEE 26th Annual Conference on. IEEE 115-125.

Author's affiliation: Department of CSE, University at Buffalo, Amherst, NY 14260 USA

Academic adviser: Kenneth. W. Regan

Correspondence address: {robertlu,regan}@buffalo.edu