Resolution of Hartmanis' Conjecture for NL-hard sparse sets

Jin-Yi Cai * D. Sivakumar †
Department of Computer Science
State University of New York
Buffalo, NY 14260
Email:{cai,sivak-d}@cs.buffalo.edu

Abstract

We resolve a conjecture of Hartmanis from 1978 about sparse hard sets for nondeterministic logspace (NL). We show that there exists a sparse hard set S for NL under logspace many-one reductions if and only if NL = L (deterministic logspace).

1 Introduction

A set is sparse if it has at most a polynomial number of strings of each length n. Sparse hard sets (and sparse complete sets) have been a fascinating subject of study in complexity theory for the past two decades. Sparse hard sets have two fundamental connections to complexity theory. First, by the fundamental result of A. Meyer [BH77], they serve as a link between nonuniform (or circuit) complexity and uniform (or Turing machine) complexity. Secondly, the various isomorphism conjectures of Berman and Hartmanis [BH77, Har78] imply that unless drastic collapses of complexity classes occur, interesting complexity classes such as NP, P, NL cannot have sparse hard or sparse complete sets. For the history and survey of interesting developments concerning sparse sets, see the articles [HOW92, You92a, You92b, CO95].

The subject of the present paper is sparse hard sets for the complexity class NL, nondeterministic logspace. In 1978, while studying the isomorphism problem for P and NL under logspace many-one reductions, Hartmanis observed that all known P-complete sets were isomorphic to each other under bijections computable (and invertible) in logspace. Motivated by this observation, Hartmanis conjectured that all P-complete sets are in fact isomorphic under logspace computable bijections. Based on similar observations about sets complete for NL, Hartmanis also conjectured that all NL-complete sets are isomorphic under logspace computable bijections. Moreover, since all the known P-complete and NL-complete sets were exponentially dense, the isomorphism conjecture implies the following conjecture about sparse complete sets:

*Research supported in part by NSF grants CCR-9057486 and CCR-9319093, and by an Alfred P. Sloan Fellowship.
†Research supported in part by NSF grant CCR-9409104.

Conjecture (Hartmanis, 1978): *There is no sparse complete set for* P *or for* NL *under logspace many-one reductions (unless* P = L *or* NL = L*, respectively).*

Clearly, if P = L (resp. if NL = L) then sparse complete sets exist for P (resp. for NL). The sparse set conjecture states that if P $\neq$ L (resp. NL $\neq$ L) then there is no P-complete (resp. NL-complete) sparse set.

The analogous conjecture for NP was settled by Mahaney in 1980 [Mah82]. The well known Karp-Lipton theorem is also concerned with sparse hard sets for NP, but under Turing reductions. They showed that if a sparse hard set exists for NP under polynomial time Turing reductions, then the Polynomial-time hierarchy collapses to its second level Σ_2^p [KL82]. Concerning the Hartmanis conjecture on sparse hard sets for P and NL, very little was known [HOT94] until the recent breakthrough by Ogihara [Ogi95], who showed that if P has a sparse hard set under logspace many-one reductions, then P $\subseteq$ DSPACE[$\log^2 n$]. The question for NL remained completely open. The conjecture of Hartmanis for P was settled by Cai and Sivakumar [CS95]. They showed that P has a sparse hard set under logspace many-one reductions iff P = L.

In this paper, we finally settle the Hartmanis conjecture for NL. We showed that there is a sparse hard set for NL under logspace many-one reductions iff NL = L. Our proof uses the algebraic techniques of [CS95]. An additional crucial ingredient in the proof is the famous result of Immerman [Imm88] and Szelepcsényi [Sze87], that NL = co-NL. Assuming the existence of a sparse hard set for NL, our proof gives a parallel algorithm for an NL-complete problem. This parallel algorithm can be implemented by a logspace-uniform circuit of polynomial-size, log-depth circuit that makes polynomially many parallel calls to the reduction from NL to the sparse set S. This implies that if NL has a sparse hard set under logspace many-one reductions, then NL = L, and if NL has a sparse hard set under (logspace-uniform) NC^1 many-one reductions, then NL = (logspace-uniform) NC^1.

2 Preliminaries

All our notations and definitions are standard. We denote by P the class of all languages recognizable in polynomial time by deterministic Turing machines. The class of all languages recognizable by deterministic Turing machines that use space no more than $O(\log n)$ is denoted LOGSPACE or L; the corresponding nondeterministic class is denoted by NL.

For circuit and parallel complexity, we use the notation SIZE-DEPTH[$s(n), d(n)$] to denote the class of languages accepted by a uniform family $\{C_n\}_{n=0}^{\infty}$ of bounded fan-in circuits of size $s(n)$ and depth $d(n)$ for inputs of length n. The criterion for uniformity of the circuit family is usually taken to mean that there is a deterministic space $(\log s(n))$-bounded transducer that, on input 0^n, outputs an encoding of the circuit C_n. The class NC^k is defined as SIZE-DEPTH[poly(n), $\log^k n$], and NC = $\bigcup_k$ NC^k. (Our NC^1 is logspace-uniform NC^1.) The following well-known relations hold among these complexity classes: NC^1 $\subseteq$ L $\subseteq$ NL $\subseteq$ NC^2.

For any language A, let $c_A(n) \doteq \|\{x \in A \mid |x| \leq n\}\|$ denote the *census function* for A. A is called (polynomially) *sparse* if $c_A(n)$ is bounded by a polynomial in n.

Given a directed graph $G = (V, E)$ and two distinguished vertices $s, t \in V$, the *s-t connectivity problem* asks whether there is a directed path from s to t in G, i.e., whether a sequence of directed edges $(s, u_1), (u_1, u_2), \ldots, (u_k, t)$ exists. The s-t connectivity problem is well-known to be complete for NL under logspace many-one reductions [Sav73]. Immerman [Imm87] has shown that this problem is complete for NL under an extremely weak form of many-one reductions called first-order projections (that are, in fact, quantifier-free). We note that the s-t connectivity problem is complete for NL (under logspace- or even NC^1-computable) many-one reductions, even if restricted to directed acyclic graphs. We call this problem "DAG-STCON." Moreover, without loss of generality, we may also assume that all instances of DAG-STCON are labeled and layered graphs, that is, graphs where all edges go from lower-numbered vertices to higher-numbered vertices. A main consequence of the completeness of DAG-STCON is that if DAG-STCON $\in \mathrm{NC}^1$, then $\mathrm{NL} = \mathrm{NC}^1$, and if DAG-STCON $\in$ L, then NL = L.

3 Main Result

Theorem 1 *If there is a sparse set S that is hard for* NL *under logspace many-one reductions, then DAG-STCON can be solved by a logspace-uniform family of polynomial size, logarithmic depth circuits that make polynomially many parallel queries to the reduction from* NL *to S.*

That is, modulo the complexity of the reduction, the parallel algorithm for DAG-STCON works in NC^1. It follows that if there is a sparse hard set for NL under many-one reductions computable in logspace-uniform NC^1, then NL equals logspace-uniform NC^1, and that if there is a sparse hard set for NL under many-one reductions computable in logspace, then NL = L.

Proof. It is known that if m is of the form $2 \cdot 3^\ell$ for some integer $\ell \geq 0$, the polynomial $x^m + x^{m/2} + 1$ is an irreducible polynomial of degree m over $GF(2)$ [vL91]. In the following, by a finite field $GF(2^m)$, where $m = 2 \cdot 3^\ell$, we refer explicitly to the field $\mathbb{Z}_2[x]/(x^m + x^{m/2} + 1)$. Following [Ogi95, CS95], we will define an auxiliary language in NL. Write $k = \binom{n}{2}$. Let B be the set of all tuples of the form $\langle G, s, t, 1^m, \alpha, \alpha^2, \ldots, \alpha^{k-1}, \beta \rangle$, where:

(1) $G = (V, E)$ is a directed, layered acyclic graph on n vertices, and has at most $k = \binom{n}{2}$ edges. Hence the adjacency matrix A_G of G is upper-triangular.

(2) s and t are vertices in G.

(3) m is of the form $2 \cdot 3^\ell$ for some integer $\ell \geq 0$.

(4) $\alpha, \beta \in GF(2^m)$.

(5) For $1 \le i < k$, α^i denotes the i-th power of α in $GF(2^m)$.

(6) $\sum_{i=0}^{k-1} \alpha^i X_{u_i v_i} = \beta$, where for $0 \le i < k$, $0 \le u_i < v_i < n$, (u_i, v_i) denotes the i-th edge in G, and $X_{u_i v_i}$ is a boolean variable that is 1 if and only if there is a path from u_i to t whose first step is the edge (u_i, v_i).

This definition is somewhat complicated by the necessity of a certain uniformity considerations. Intuitively it may appear that in place of $\langle G, s, t, 1^m, \alpha, \alpha^2, \ldots, \alpha^{k-1}, \beta\rangle$, we should have been able to use just $\langle G, s, t, 1^m, \alpha, \beta\rangle$. Unfortunately this is not so simple. However, for readability, we will abbreviate $\langle G, s, t, 1^m, \alpha, \alpha^2, \ldots, \alpha^{k-1}, \beta\rangle$ by $\langle G, s, t, 1^m, \alpha, \beta\rangle$ throughout this paper.

Claim 2 $B \in \mathrm{NL}$.

Proof of Claim. We will build a nondeterministic logspace machine N that accepts B. First we argue that, in $O(\log n + \log m)$ space, N may deterministically verify that the values $\alpha^2, \alpha^3, \ldots, \alpha^{k-1}$ are indeed the correct powers of α. To do this, N proceeds sequentially, for i from 1 up to $k-2$, verifying the validity of $\alpha \cdot \alpha^i = \alpha^{i+1}$. For a fixed i, N accomplishes this by sequentially computing each bit of α^{i+1} from the values of α and α^i given in the input, and checking it against the value of α^{i+1} given in the input. This requires two counters, one that can count up to $k-2$ and one that can count up to m. The counters can be implemented in space $O(\log n + \log m)$. Now to check the bits: For $\gamma \in GF(2^m)$, let $(\gamma)_j$ denote the j-th bit of γ, and let $P_\gamma \in \mathbb{Z}_2[x]$ denote the polynomial whose coefficients are given by the bits of γ. Thus, $\alpha^{i+1} = (P_\alpha \cdot P_{\alpha^i}) \bmod (x^m + x^{m/2} + 1)$, where P_α and P_{α^i} are multiplied in the ring $\mathbb{Z}_2[x]$. For $0 \le j \le 2m-2$, the coefficient of x^j in $(P_\alpha \cdot P_{\alpha^i})$ is given by $\sum_{\sigma+\tau=j} (\alpha)_\sigma (\alpha^i)_\tau$. This is a mod 2 sum of at most m bits. Denote this sum by $S(j)$. When the product $(P_\alpha \cdot P_{\alpha^i})$ is reduced modulo $x^m + x^{m/2} + 1$, the j-th bit of α^{i+1}, for $0 \le j < m$, is given by the sum, in $\mathbb{Z}_2$, of the following four contributions $S_1(j), S_2(j), S_3(j), S_4(j)$.

(1) For $0 \le j < m$, $S_1(j) = S(j)$. This contribution comes from the term x^j of the product $(P_\alpha \cdot P_{\alpha^i})$.

(2) For $0 \le j < m/2$, $S_2(j) = S(j+m)$. This contribution comes from the term x^τ, where $m \le \tau < 3m/2$, of the product $(P_\alpha \cdot P_{\alpha^i})$.

(3) For $m/2 \le j < m$, $S_3(j) = S(j + m/2)$. This contribution also comes from the term x^τ, where $m \le \tau < 3m/2$, of the product $(P_\alpha \cdot P_{\alpha^i})$.

(4) For $0 \le j < m/2$, $S_4(j) = S(j + 3m/2)$. This sum equals the coefficient of the term $x^{m+m/2+j}$ of the product $(P_\alpha \cdot P_{\alpha^i})$, which is equal, mod $x^m + x^{m/2} + 1$, to x^j.

Clearly, each of these sums can be evaluated in space $O(\log m)$.

Now, we may assume that the input is legitimate, that is, all the powers of α are correctly presented. Testing whether $\langle G, s, t, 1^m, \alpha, \beta\rangle \in B$ requires computing polynomially many predicates X_{uv}. The language $Z = \{\langle G, s, t, u, v\rangle \mid X_{uv} = 1\}$ is in NL, and since NL = co-NL, its complement Z^c is also in NL. The nondeterministic logspace machines for Z and Z^c can be used to build a nondeterministic logspace machine that computes X_{uv} in the following strong sense: every computation either outputs the correct value of X_{uv} or aborts in a "DON'T KNOW" state, and at least one computation is guaranteed to output the correct value of X_{uv}.

Using this, we will build the nondeterministic $O(\log n + \log m)$ space-bounded machine N that accepts B as follows: Since the elements of the field $GF(2^m)$ have m-bit representations, the machine N cannot write down entries of the field explicitly in its workspace during the computation to check $\sum_{i=0}^{k-1} \alpha^i X_{u_i v_i} = \beta$. Instead, it maintains a $(\log m)$-bit counter that checks, bit by bit, if the above equality holds. To check the equality of the j-th bit of $\sum_{i=0}^{k-1} \alpha^i X_{u_i v_i}$ and β, the machine N proceeds as follows: N first initializes a bit $b_j = 0$. Then, sequentially and nondeterministically N computes $X_{u_i v_i}$ for each edge (u_i, v_i). If $X_{u_i v_i} = 0$, it goes on to compute the next value $X_{u_{i+1} v_{i+1}}$. If $X_{u_i v_i} = 1$, then it finds the j-th bit $(\alpha^i)_j$ of α^i (which is present in the input), and updates $b_j = b_j \oplus (\alpha^i)_j$. Notice that, by design, every computation path of N either computes $X_{u_i v_i}$ correctly (with a "certificate") and proceeds, or it aborts. Finally, N accepts $\langle G, s, t, 1^m, \alpha, \beta\rangle$ if and only if for all j, the j-th bits of $\sum_{i=0}^{k-1} \alpha^i X_{u_i v_i}$ and β match. *End of Proof of Claim.*

By hypothesis, $B \leq_m S$. Let f denote the (logspace- or NC^1-computable) function that reduces B to S. We will show how to solve DAG-STCON using f as an oracle. Fix $G = (V, E)$, s and t. Let $n = |V|$ and $k = \binom{n}{2}$. Clearly $|\langle G, s, t, 1^m, \alpha, \beta\rangle|$ is bounded polynomially in n and m. If f is a logspace-computable function that reduces B to S, the bound on the length of queries made by f on inputs of length $|\langle G, s, t, 1^m, \alpha, \beta\rangle|$ is some polynomial $q(n, m)$. Let $p(n, m)$ be a polynomial that bounds the number of strings in S of length at most $q(n, m)$. We will choose the smallest m of the form $2 \cdot 3^\ell$ such that $2^m / p(n, m) \geq k = \binom{n}{2}$. It is clear that $m = O(\log n)$. Let $\mathbb{F}$ denote the finite extension $GF(2^m)$ of $GF(2)$. Some basic facts about computing in the field $\mathbb{F}$ are summarized in the Appendix.

Our parallel algorithm for DAG-STCON begins by computing $f(\langle G, s, t, 1^m, \alpha, \beta\rangle)$ for all $\alpha, \beta \in \mathbb{F}$. (Setting up the required powers of α is an easily accomplished task, since it can be precomputed off-line in logspace; see Appendix.) For every $\alpha \in \mathbb{F}$, there is a unique element $\beta_\alpha \in \mathbb{F}$ such that $\langle G, s, t, 1^m, \alpha, \beta_\alpha\rangle \in B$, and therefore f maps precisely one tuple of the form $\langle G, s, t, 1^m, \alpha, \beta\rangle$ into S. Since $2^m/p(n, m) \geq k$, there is at least one string $w^* \in S$ such that the number of α satisfying $f(\langle G, s, t, 1^m, \alpha, \beta_\alpha\rangle) = w^*$ is at least k. Strings w that have $\geq k$ pre-images α will be called *popular*.

For any w, whenever $f(\langle G, s, t, 1^m, \alpha, \beta\rangle) = w$, under the *assumption* that $w \in S$ we have an equation

$$1\ X_{u_0 v_0} + \alpha X_{u_1 v_1} + \alpha^2 X_{u_2 v_2} + \ldots + \alpha^{k-1} X_{u_{k-1} v_{k-1}} = \beta$$

in the variables X_{uv}. Thus for every *popular* w, we will have a *system* of at least k such equations; moreover, the system of equations is *correct* if and only if $w \in S$. Of course, there could be many popular w, and we don't know which ones are in S. To handle this, we will *assume* that every popular w is a string in S, and attempt to solve for the X_{uv}'s for all $u, v \in V, u < v$. This scheme produces a polynomial number of sets of solutions. As long as there is at least one popular $w^* \in S$, one of our assumptions must be correct, and we will have the correct solution. It remains, therefore, to show: (1) how to *solve* the systems of equations by a poly-size log-depth circuit, and (2) how to *verify* the correctness of solutions.

Solution of the Systems of Equations.

For every popular w, when the equations produced are written as matrix-vector product of the form $\mathcal{A}X = \mathcal{B}$, the $k \times k$ matrix $\mathcal{A}$ obtained is a *Vandermonde* matrix. Moreover, since the α's are distinct, the matrix $\mathcal{A}$ has full rank over $\mathbb{F}$. The following lemma, which is an essential part of the proof for the main result of [CS95], shows that this system of equations can be solved in NC^1. (For completeness we will sketch the proof of the lemma in the Appendix.)

Lemma 3 *Let $\mathbb{K} = GF(2^M)$, where $M = O(\log N)$, and M is of the form $2 \cdot 3^L$ for some integer $L \geq 0$. Solving a system $AX = B$ of N equations in N unknowns over the field $\mathbb{K}$, where A is a Vandermonde matrix of full rank over $\mathbb{K}$, can be done by an $O(\log N)$-space uniform circuit of size $N^{O(1)}$ and depth $O(\log N)$.*

Verification of the Solutions.

Each set of solutions for the X_{uv}'s can be assumed to be in the form of a matrix X of the same dimensions as A_G, the adjacency matrix of G. By its definition, X_{uv} is 1 only if $(u, v) \in E$. Therefore, for each candidate set of solutions, we will first verify that the condition $X_{uv} \leq A_G(u, v)$ holds for all $u, v \in V$. It is easy to see that this test can be performed simultaneously on all sets of solutions by a polynomial-size, log-depth circuit. Note that since A_G is a strictly upper-triangular matrix, every X that passes this test is strictly upper-triangular.

For every $u \in V$, we first compute the boolean variable X_u that is 1 if and only if $u = t$ or there exists some v such that $X_{uv} = 1$. In matrix terms, $X_u = 1$ if and only if $u = t$ or there is at least one 1 in the row corresponding to u in the matrix X. This computation can be easily done by a polynomial-size, log-depth circuit, since it only requires computing the OR of n bits. Next we perform the following *local consistency test*: for every u such that $X_u = 1$ and for every v such that $X_{uv} = 1$, verify that $X_v = 1$. This test ensures that if X promises a path from u to t with v as the first vertex, then indeed X also promises some path from v to t. Notice that the latter path cannot include the vertex u since G is acyclic. This is important because, otherwise, it is possible that X passes this test by setting $X_{uv} = X_{vu} = 1$ even though t is reachable from neither of u and v. It is clear that the local consistency test can be performed by a polynomial-size log-depth circuit.

Finally we argue that there exists a path from s to t if and only if some set of solutions

X which passes all these tests and has $X_s = 1$. Clearly, if there is a path from s to t then the correct solution for X will pass all the tests and have $X_s = 1$.

Next we claim that if X passes all the tests, then $X_z = 0$ for all $z > t$. The claim is vacuous if t is the last vertex. Otherwise, we prove the claim by induction, starting from the last vertex. By the first test against the adjacency matrix and since the graph is layered, the base case is clear. Assume inductively for some $t < z_0$, that $X_z = 0$ for all z such that $z_0 < z$. If $X_{z_0} = 1$, then by the definition of X_{z_0}, for some $z > z_0$, $X_{z_0 z} = 1$. However, it then fails the *local consistency test*, since $X_z = 0$.

It follows in particular that if $s > t$ and X passes all the tests, then $X_s = 0$.

To complete the proof, suppose that X passes all the tests. We argue that whenever $X_u = 1$ for some $u \leq t$, there is a path from u to t. The base case, namely $u = t$, is trivial. Suppose $X_u = 1$ for some $u < t$. By definition, there is a vertex v such that $X_{uv} = 1$ and $X_v = 1$. The first test ensures that there is an edge (u, v). If $v = t$, it is clear that there is a path from u to t. If $v < t$, by the inductive hypothesis, there is a path from v to t, which, together with the edge (u, v), gives a path from u to t. $\square$

Corollary 4 *There is no sparse hard set for* NL *under* L*-many-one reductions iff* $\mathrm{NL} \neq \mathrm{L}$.

The proofs of the next two theorems combine the above technique with ideas from [CNS95] and from [CNS95, Mel95], respectively.

Theorem 5 *If there is a sparse hard set for* NL *under logspace-computable randomized many-one reductions with two-sided error, then* $\mathrm{NL} = \mathrm{RL}$*, where* RL *is the class of languages accepted by logspace Turing machines with two-way access to the random tape. If there is a sparse hard set for* NL *under randomized many-one reductions with two-sided error that are computable in logspace-uniform* NC^1*, then* $\mathrm{NL} \subseteq \mathrm{RNC}^1$.

Theorem 6 *If there is a sparse hard set for* NL *under logspace bounded truth-table reductions, then* $\mathrm{NL} = \mathrm{L}$*. If there is a sparse hard set for* NL *under bounded truth-table reductions computable in logspace-uniform* NC^1*, then* NL *equals logspace-uniform* NC^1.

References

[BH77] L. Berman and J. Hartmanis. On isomorphisms and density of NP and other complete sets. *SIAM Journal on Computing*, 6:305–321, 1977. A preliminary version appeared in STOC 1976.

[BvzGH82] A. Borodin, J. von zur Gathen, and J. Hopcroft. Fast parallel matrix and GCD computations. *Information and Control*, 52:241–256, 1982.

[CNS95] J. Cai, A. Naik, and D. Sivakumar. On the existence of hard sparse sets under weak reductions. In the proceedings of *The 13th Annual Symposium on Theoretical Aspects of Computer Science* (STACS), 1996, 307–318.

[CO95] J. Cai and M. Ogihara. Sparse sets versus complexity classes. UBCS–TR 95-41, Computer Science Dept., University at Buffalo, September 1995.

[CS95] J. Cai and D. Sivakumar. The resolution of a Hartmanis conjecture. In *Proc. 36th Annual IEEE Symposium on Foundations of Computer Science*, pages 362–373, 1995.

[Har78] J. Hartmanis. On log-tape isomorphisms of complete sets. *Theoretical Computer Science*, 7(3):273–286, 1978.

[HOT94] L. Hemachandra, M. Ogiwara, and S. Toda. Space-efficient recognition of sparse self-reducible languages. *Computational Complexity*, 4:262–296, 1994.

[HOW92] L. Hemachandra, M. Ogiwara, and O. Watanabe. How hard are sparse sets. In *Proc. 7th Annual IEEE Conference on Structure in Complexity Theory*, pages 222–238, 1992.

[Imm87] N. Immerman. Languages that capture complexity classes. *SIAM Journal on Computing*, 16(4):760–778, 1987.

[Imm88] N. Immerman. Nondeterministic space is closed under complementation. *SIAM Journal on Computing*, 17:935–938, 1988.

[KL82] R. Karp and R. Lipton. Turing machines that take advice. *L'enseignement Mathematique*, 28(3/4):191–209, 1982.

[Mah82] S. Mahaney. Sparse complete sets for NP: Solution of a conjecture of Berman and Hartmanis. *J. Comput. System Sci.*, 25(2):130–143, 1982.

[Mel95] D. Van Melkebeek. On reductions of P sets to sparse sets. TR 95-06, Computer Science Dept., The University of Chicago, 1995.

[Ogi95] M. Ogihara. Sparse hard sets for P yield space-efficient algorithms. In *Proc. 36th Annual IEEE Symposium on Foundations of Computer Science*, pages 354–361, 1995.

[Sav73] W. Savitch. Maze recognizing automata and nondeterministic tape complexity. *J. Comp. Sys. Sci.*, 7:389–403, 1973.

[Sze87] R. Szelepcsényi. The method of forcing for nondeterministic automata. *Bull. of the EATCS*, 33:96–100, 1987.

[vL91] J.H. van Lint. *Introduction to Coding Theory.* Springer-Verlag, 1991.

[VV86] L. Valiant and V. Vazirani. NP is as easy as detecting unique solutions. *Theoretical Computer Science*, 47:85–93, 1986.

[You92a] P. Young. How reductions to sparse sets collapse the polynomial-time hierarchy: A primer (Part I). *SIGACT News*, 23(3):107–117, 1992.

[You92b] P. Young. How reductions to sparse sets collapse the polynomial-time hierarchy: A primer (Part II). *SIGACT News*, 23(4):83–94, 1992.

Appendix

Computation in $\mathbb{F}$: First we recall that the polynomial $X^{2\cdot 3^\ell} + X^{3^\ell} + 1 \in \mathbb{Z}_2[X]$ is an irreducible polynomial over $\mathbb{Z}_2$ for all $\ell \geq 0$ [vL91], and collect some facts about implementing the basic operations of $\mathbb{F}$. For each operation, the number of processors needed is at most $n^{O(1)}$.

(1) Finding a primitive element ω that generates the multiplicative group $\mathbb{F}^*$ of $\mathbb{F}$ can be done in logspace by exhaustive search.

(2) Adding two elements $y_1, y_2 \in \mathbb{F}$ is just the bitwise exclusive-or of the representations of y_1 and y_2, and can be done in depth $O(1)$. Adding $n^{O(1)}$-many elements can be done in depth $O(\log n)$.

(3) Multiplying two elements of $\mathbb{F}$ (in the straightforward way) can be done by a circuit of depth $O(\log\log n)$ and size $(\log n)^{O(1)}$.

(4) Raising the generator ω to any power $i < 2^m$, or computing the discrete logarithm of any element with respect to ω, can be done by table lookup in depth $O(\log n)$. The tables themselves can be precomputed using $O(\log n)$ space.

(5) Multiplying $k = n^{O(1)}$ elements of $\mathbb{F}$ can be done in $O(\log n)$ depth. The idea is to use the discrete logarithms of the k elements with respect to the generator ω, and convert multiplications to additions of k $O(\log n)$-bit integers (modulo $2^m - 1$), which can be done in $O(\log n)$ depth using the folklore 3-to-2 trick.

Proof of Lemma 3

Lemma 3 *Let $\mathbb{F} = GF(2^m)$, where $m = O(\log n)$, and m is of the form $2 \cdot 3^\ell$ for some integer $\ell \geq 0$. Solving a system $Ax = b$ of n equations in n unknowns over the field $\mathbb{F}$, where A is a Vandermonde matrix of full rank over $\mathbb{F}$, can be done by an $O(\log n)$-space uniform circuit of size $n^{O(1)}$ and depth $O(\log n)$.*

Proof. Whereas solving general linear equation systems seems to require NC^2 [BvzGH82], we will arrive at our NC^1 solution via closed formulae. Observe that an equation of the form $\sum_{j=0}^{n-1} g_j u^j = v$ can be viewed as specifying the value of the polynomial $G(u) \doteq \sum_{j=0}^{n-1} g_j u^j$ at the point $u \in \mathbb{F}$. With this viewpoint, our task is to compute the polynomial G, that is, to find the coefficients g_j of G. Clearly if we can evaluate $G(u)$ at n distinct points $u_1, \ldots, u_n \in \mathbb{F}$, then we an recover the coefficients g_j by Lagrange interpolation as follows:

$$G(u) = \sum_{i=1}^{n} G(u_i) Q_i = \sum_{i=1}^{n} v_i Q_i$$

where
$$Q_i = \frac{(u-u_1)\dots(u-u_{i-1})(u-u_{i+1})\dots(u-u_n)}{(u_i-u_1)\dots(u_i-u_{i-1})(u_i-u_{i+1})\dots(u_i-u_n)} = \prod_{k\neq i}\frac{(u-u_k)}{(u_i-u_k)}.$$
For $0 \le j < n$, g_j is the coefficient of u^j in $G(u)$. Collecting the terms corresponding to u^j, we have
$$g_j = \sum_{i=1}^{n} \frac{(-1)^{1+i}v_i}{\prod_{k\neq i}(u_k-u_i)} P_{n-j-1}(u_1,\dots,\widehat{u_i},\dots,u_n).$$
Here $\widehat{u_i}$ denotes that u_i is missing from the list $u_1,\dots,u_n$, and P_k denotes the k-th elementary symmetric polynomial, defined as follows:
$$P_0(y_1,\dots,y_\ell) = 1; \qquad P_k(y_1,\dots,y_\ell) = \sum_{\substack{I\subseteq[\ell]\\|I|=k}} \prod_{i\in I} y_i, \qquad k>0.$$

By Facts (3) and (5), computing $v_i/(\prod_{k\neq i}(u_k-u_i))$ in NC^1 is fairly straightforward. Hence it suffices to show how to compute the polynomials $P_k(u_1,\dots,\widehat{u_i},\dots,u_n)$, in logspace-uniform NC^1. A folklore theorem indicates that this can be done in non-uniform NC^1. For our application, however, the uniformity is crucial.

It is easy to see that for $y_1,\dots,y_\ell \in \mathbb{F}$, $P_k(y_1,\dots,y_\ell)$ equals $P_k(y_1,y_2,\dots,y_\ell,0,0,\dots,0)$ for any number of extra zeroes. Let $r = |\mathbb{F}^*|$, the number of elements in the multiplicative group of $\mathbb{F}$. We will give an NC^1 algorithm to compute the elementary symmetric polynomial of r elements, not necessarily distinct, from the finite field $\mathbb{F}$. By appending $r-\ell$ zeroes, we can then compute $P_k(y_1,y_2,\dots,y_\ell)$.

For $0 < k \le r$, the value of the elementary symmetric polynomial $P_k(y_1,y_2,\dots,y_r)$ is the coefficient of X^{r-k} in $h(X) \doteq \prod_{i=1}^{r}(X+y_i) - X^r$. Note that, given any $\alpha \in \mathbb{F}$, $h(\alpha)$ can be evaluated in NC^1, by Facts (2) and (5).

If we write $h(X)$ as $\sum_{i=0}^{r-1} a_i X^i$, the coefficient $a_i = P_{r-i}(y_1,\dots,y_r)$ for $0 \le i < r$. The idea now is to choose α's carefully from $\mathbb{F}$, compute $h(\alpha)$ and compute the coefficients a_i by interpolation. If we choose ω to be a primitive element of order r in $\mathbb{F}^*$, the powers of ω, namely $1=\omega^0,\omega^1,\omega^2,\dots,\omega^{r-1}$, run through the elements of $\mathbb{F}^*$. For $0 \le i < r$, let $b_i = h(\omega^i)$. The relationship between the pointwise values (b_i's) and the coefficients (a_i's) of $h(X)$ can be written as:
$$\begin{pmatrix} b_0 \\ b_1 \\ \vdots \\ b_{r-1} \end{pmatrix} = \begin{pmatrix} 1 & \omega^0 & \omega^{0\cdot 2} & \dots & \omega^{0\cdot(r-1)} \\ 1 & \omega^1 & \omega^{1\cdot 2} & \dots & \omega^{1\cdot(r-1)} \\ \vdots & \vdots & \vdots & & \vdots \\ 1 & \omega^{r-1} & \omega^{(r-1)\cdot 2} & \dots & \omega^{(r-1)\cdot(r-1)} \end{pmatrix} \begin{pmatrix} a_0 \\ a_1 \\ \vdots \\ a_{r-1} \end{pmatrix}.$$
The above matrix, which we will denote by Ω, is the Discrete Fourier Transform matrix, and is a Vandermonde matrix. Since the powers of ω are all distinct, Ω is invertible, and one can compute the coefficients a_i by $(a_0,\dots,a_{r-1})^T = \Omega^{-1}(b_0,\dots,b_{r-1})^T$. The crucial advantage over the earlier Vandermonde system is that with this particular choice of Ω, the matrix Ω^{-1} has a simple explicit form: $\Omega^{-1}_{ij} = 1/(\Omega_{ij}) = \omega^{-ij}$. Computing the coefficients of $h(X)$ is now simply a matrix-vector multiplication. This completes the proof of the lemma. $\square$