On the Power of Parity Polynomial Time

Jin-yi Cai[1]
Lane A. Hemachandra[2]

December, 1987

CUCS-274-87

[1]Department of Computer Science, Yale University, New Haven, CT 06520
[2]Department of Computer Science, Columbia University, New York, NY 10027

On the Power of Parity Polynomial Time

Jin-yi Cai[*]
Department of Computer Science
Yale University
New Haven, CT 06520

Lane A. Hemachandra[†]
Department of Computer Science
Columbia University
New York, NY 10027

December, 1987

Abstract

This paper proves that the complexity class $\oplus$P, parity polynomial time [PZ83], contains the class of languages accepted by NP machines with few accepting paths. Indeed, $\oplus$P contains a broad class of languages accepted by path-restricted nondeterministic machines.

In particular, $\oplus$P contains the polynomial accepting path versions of NP, of the counting hierarchy, and of Mod_mNP for $m > 1$.

We further prove that the class of nondeterministic path-restricted languages is closed under bounded truth-table reductions.

1 Introduction and Overview

One of the goals of computational complexity theory is to classify the inclusions and separations of complexity classes. Though nontrivial separation of complexity classes is often a challenging problem (e.g. P $\neq$ NP?), the inclusion structure of complexity classes is progressively becoming clearer [Sim77,Lau83,Zac86,Sch87]. This paper proves that $\oplus$P contains a broad range of complexity classes.

1.1 Parity Polynomial Time

The class $\oplus$P, parity polynomial time, was defined and studied by Papadimitriou and Zachos as a "moderate version" of Valiant's counting class #P.

[*]Research supported by NSF grant CCR-8709818.
[†]Research supported in part by a Hewlett-Packard Corporation equipment grant.

Definition 1.1 [PZ83] $\oplus$P = $\{L \mid$ there is a nondeterministic polynomial-time Turing machine N such that $x \in L$ if and only if $N(x)$ has an odd number of accepting paths$\}$.

Papadimitriou and Zachos presented a natural complete language and proved that $\oplus P^{\oplus P} = \oplus P$, thus showing that $\oplus$P seems to behave differently than NP. They left as an open problem the relationship between $\oplus$P and NP. Though the problem remains open, this paper shows that $\oplus$P contains broad subclasses of NP as well as classes that are thought not to be contained in NP.

More recently, $\oplus$P has become intimately connected with the theory of near-testable sets—sets A for which one can test in polynomial time whether $(x \in A) \oplus (x^+ \in A)$, where x^+ indicates the lexicographical successor of x [GJY87]. Over the last year, the class of near-testable sets, NT, has been proven to share the polynomial many-one degree of $\oplus$P [GHJY87], and it follows from this and the work of Bennett and Gill [BG81] that $P^A \neq NT^A$ with probability one relative to a random oracle A [Hem87a,GHJY87].

1.2 Classes with at Most Polynomially Many Accepting Paths

Valiant first introduced the notion of nondeterministic polynomial-time Turing machines with bounded numbers of accepting computations [Val76]—he defined UP, unique polynomial time, to be the class of languages accepted by nondeterministic polynomial-time Turing machines that never have more than one accepting path. Grollmann and Selman showed that P $\neq$ UP if and only if one-way functions exist, thus UP is a class central to cryptography [GS84]. Recently, the possibility has been raised that UP lacks complete languages [HH86], and it has been conjectured [JY85] and refuted in relativized worlds [HH87] that P = UP if and only if all NP-complete sets are p-isomorphic.

Allender's class FewNP, a class that contains UP, has the property that P = FewNP if and only if all sparse sets in P are P-printable [All86]. FewNP is also related to the existence and invertibility of poly-to-one one-way functions [All85,All86].

More generally, we define the class Few to capture the general notion of polynomial-path nondeterminism.

Definition 1.2 Few is the class of all languages L such that there is a nondeterministic polynomial-time Turing machine N, a polynomial-time computable predicate $Q(\cdot, \cdot)$, and a polynomial $q(\cdot)$, such that:

1. $x \in L$ if and only if $Q(x, ||N(x)||)$ and

2. $(\forall x)\,[||N(x)|| \leq q(|x|)]$,

where $||N(x)||$ denotes the number of accepting paths of $N(x)$.

In words, a language L is in Few if there is a nondeterministic polynomial-time Turing machine N that never has many accepting paths, and a polynomial-time computable predicate Q, such that on each input x, Q can look at x and the number of accepting paths of $N(x)$ and determine if $x \in L$.

It is immediate that the class Few contains the common path-restricted classes. In particular Few contains FewNP, UP, and FewCH—the polynomial accepting path version of the counting hierarchy [Wec85,CGH*86,GW87]. Indeed, FewCH is the subclass of Few such that for some finite or cofinite set S, $Q(x, k)$ is true exactly when $k \in S$. Note that in the general case $Q(x, k)$ may depend on x.

Section 2 presents a proof that $\oplus$P $\supseteq$ Few, from which it follows that $\oplus$P contains FewNP and FewCH.

Section 3 proves that Few is closed under bounded truth-table reductions. This closure result shows behavior different from that of many standard complexity classes.

2 $\oplus$P $\supseteq$ Few

2.1 Results and Proof

This section shows that $\oplus$P $\supseteq$ Few. Given a language $L \in$ Few, our proof shows how to construct a $\oplus$P machine that accepts L. The parity machine will be built using as components the machine N and the polynomial predicate Q that demonstrate (according to Definition 1.2) that $L \in$ Few.

Theorem 2.1 $\oplus$P $\supseteq$ Few.

The idea of the proof of Theorem 2.1 is that, given a language in Few, we can construct a $\oplus$P machine that dynamically organizes its own actions to insure that it has an odd number of accepting paths exactly when the acceptance predicate of the Few machine demands acceptance. Since Few acceptance predicates are sensitive to the input string, the form of the computation tree that our $\oplus$P machine creates will vary depending on the input.

Corollary 2.2

1. $\oplus$P $\supseteq$ FewNP.

2. $\oplus P \supseteq$ FewCH.

3. $\oplus P \supseteq \text{FewMod}_m NP$, for all m.[1]

Corollary 2.2 follows from Theorem 2.1 and the fact that Few $\supseteq$ FewNP, Few $\supseteq$ FewCH, and Few $\supseteq \text{FewMod}_m NP$.

Proof of Theorem 2.1 We wish to show that an arbitrary language $L \in$ Few is in $\oplus P$. Let N, $Q(\cdot, \cdot)$, and $q(\cdot)$ be the machine, predicate, and polynomial that, according to Definition 1.2, place $L \in$ Few. We will use these to construct a $\oplus P$ machine $M_\oplus$ (a nondeterministic machine with the parity accepting mechanism) that accepts L.

On input x, $M_\oplus$ will determine a sequence $a_0, a_1, \ldots, a_{q(|x|)}$ of numbers, each zero or one. We will specify later how to use Q to determine these numbers. Then, nondeterministically, $M_\oplus$ will do the following for each $0 \leq j \leq q(|x|)$:

if $a_j = 0$: do nothing

if $a_j = 1$: nondeterministically for every j-tuple of computation paths of $N(x)$ do:

> **if all j paths in the tuple accept:** accept
>
> **otherwise:** reject.

Note that the "accept" and "reject" above refer only to a specific nondeterministic path of our $\oplus P$ machine $M_\oplus$. By convention we consider every machine to have exactly one 0-tuple of accepting paths. That is, we take the convention that $\binom{0}{0} = 1$.

How many accepting paths will $M_\oplus$ have on input x? If $N(x)$ has m accepting computations, the above procedure insures that the number of accepting paths of $M_\oplus(x)$ is exactly:

$$\sum_{0 \leq i \leq q(|x|)} a_i \binom{m}{i}.$$

Define this sum to be $s(m)$. We are done if we can effectively (i.e., in polynomial time) compute $a_0, \ldots, a_{q(|x|)}$ so that:

$$(\forall m : 0 \leq m \leq q(|x|))[s(m) \text{ is odd} \iff Q(x, m) \text{ is true}].$$

Let us see how to choose the $\{a_i\}$.

[1] $\text{Mod}_m NP$ is the class of languages accepted by nondeterministic polynomial-time Turing machines that by definition accept if and only if their number of accepting paths is congruent to one mod m. "Few" indicates a polynomial bound on the number of accepting computations.

Step 0: Compute in polynomial time $Q(x, 0)$. If $Q(x, 0)$ is true, then set $a_0 = 1$, otherwise set $a_0 = 0$. Note that if $||N(x)||$, the number of accepting paths of $N(x)$, is zero, then our $\oplus$P machine $M_\oplus(x)$ will accept exactly when $x \in L$.

$$\vdots$$

Step k ($0 < k \leq q(|x|)$) : At this point, we have chosen $a_0, \ldots, a_{k-1}$ to insure that $(\forall m : 0 \leq m \leq k-1)[s(m)$ is odd $\iff Q(x, m)$ is true]. That is, if the number of accepting paths of $N(x)$ is at most $k-1$, then $M_\oplus(x)$ will display the correct behavior.

At this step, we will set a_k to insure that $M_\oplus(x)$ will act correctly when $N(x)$ has exactly k accepting paths.

By convention, we represent true by the value 1 and false by the value 0; thus, predicate $Q(x, k)$ has value 0 or 1. Compute in polynomial time $Q(x, k)$, and compute (easily in polynomial time)

$$t =_{\text{def}} \sum_{0 \leq i \leq k-1} a_i \binom{k}{i}.$$

Case 1: If $Q(x, k) \equiv t \pmod 2$, then set $a_k = 0$. We have assured that $s(k)$ is odd if and only if $Q(x, k)$ is true.

Case 2: In this case, $Q(x, k) \not\equiv t \pmod 2$, thus $Q(x, k) - t \equiv 1 \pmod 2$. Crucially, the coefficient of a_k is $\binom{\text{number of paths}}{k}$. However, when there are exactly k paths, the coefficient is $\binom{k}{k} = 1$. Thus, by choosing $a_k = 1$, we change the sum by *exactly* one:

$$t + 1 = \left(\sum_{0 \leq i \leq k-1} a_i \binom{k}{i} \right) + 1 = \sum_{0 \leq i \leq k} a_i \binom{k}{i} = s(k).$$

Thus, by setting $a_k = 1$, we have insured that $Q(x, k) - s(k) \equiv Q(x, k) - (t + 1) \equiv 0 \pmod 2$. So, again, $s(k)$ is odd if and only if $Q(x, k)$ is true.

The above procedure for choosing the $\{a_i\}$ can be done in polynomial time. Each of the polynomial ($q(|x|)$) number of steps takes polynomial time as Q is a polynomial predicate and the binomial coefficients require a polynomial number of multiplications and divisions since we deal only with values of $k \leq q(|x|)$.

Thus, we have defined a $\oplus$P machine, $M_\oplus$, that accepts L, an arbitrarily chosen language from Few. □

For many predicates Q corresponding to natural complexity classes, the set $\{a_i\}$ of the above proof has a simple form. For the case of FewNP—$Q(x, k)$ is true if and only if $k > 0$—we have $a_0 = 0$ and $a_i = 1$ for $i > 0$. This is clear as 0 is even and $\sum_{0 < j \leq k} \binom{k}{j} = 2^k - 1$

is odd, $k > 0$. US [BG82] is the class characterized by nondeterministic polynomial-time Turing machines that accept when they have exactly one accepting path (and reject if the number of accepting paths is zero or greater than one). For the polynomially-bounded accepting path version of US, FewUS, we have $a_i = i \bmod 2$.

2.2 Generalizations

Note that at the crucial step in the proof of Theorem 2.1, a binomial coefficient collapsed to one, allowing us to change the parity at will. There is nothing special about the mod 2 underlying $\oplus$P. If we choose our a_i's from $\{0, 1, \ldots, m-1\}$, $m \geq 2$, we can show that $\mathrm{Mod}_m\mathrm{NP} \supseteq$ Few by repeating the above proof with slight modifications.

Theorem 2.3 For each $m > 1$, $\mathrm{Mod}_m\mathrm{NP} \supseteq$ Few.

It follows immediately from Theorem 2.1, Corollary 2.2, and the closure of $\oplus$P under Turing reductions [PZ83] that:

Corollary 2.4

1. $\oplus\mathrm{P} \supseteq \mathrm{P}^{\mathrm{Few}}$.
2. $\oplus\mathrm{P} \supseteq \mathrm{P}^{\mathrm{FewNP}}$.
3. $\oplus\mathrm{P} \supseteq \mathrm{P}^{\mathrm{FewCH}}$, where CH is the counting hierarchy (see page 3).
4. $\oplus\mathrm{P} \supseteq \mathrm{P}^{\mathrm{FewMod}_m\mathrm{NP}}$, for all m.

3 Few Is Closed Under Bounded Truth-Table Reductions

$A \leq^p_{j-tt} B$ ("A j-truth-table reduces to B") if there is a polynomial-time machine M such that $M(x)$ answers the question "$x \in A$?" by printing a list of at most j questions to B, which are then simultaneously answered, after which M must determine if $x \in A$ with at most polynomially more computation time [LLS75]. We say that A bounded truth-table reduces to B ($A \leq^p_{btt}$) if there is a constant j such that $A \leq^p_{j-tt} B$. This notion of polynomial-time bounded truth-table reductions has been used by Ukkonen [Ukk83] and Yesha [Yes83] to strengthen Mahaney's Theorem [Mah82]. Throughout this paper, we consider this *polynomial time* version of bounded truth-table reducibility, which is common in computational complexity theory, and simply refer to it as bounded truth-table reducibility.

This section proves that Few is closed under bounded truth-table reductions.

Lemma 3.1 $\bigcup_k \mathrm{P}^{\mathrm{Few}[k]} = \{L \mid L \leq^p_{btt} \mathrm{Few}\}$, where $[k]$ indicates that on each input at most k oracle calls are made.

Lemma 3.1 follows immediately from the fact that $\mathrm{P}^{\mathrm{Few}[m]} \subseteq \{L \mid L \leq^p_{2^m-tt} \mathrm{Few}\} \subseteq \mathrm{P}^{\mathrm{Few}[2^m]}$, so these two bounded hierarchies are interleaved.

Theorem 3.2 Few is closed under bounded truth-table reductions:

$$\mathrm{Few} = \bigcup_k \mathrm{P}^{\mathrm{Few}[k]}.$$

Corollary 3.3 Few is closed under union, intersection, and complementation.

Corollary 3.3 follows immediately from Theorem 3.2, using appropriate truth tables.

The work that remains in this section lies in the proof of Theorem 3.2. The proof's strategy is the following. Suppose we know that $L \leq^p_{btt} M$ and $M \in \mathrm{Few}$. We use the bounded truth-table reduction to get a constant number of queries to M; we combine these queries into the action of a single new nondeterministic machine whose paths encode the information of all the component Few queries and whose accepting paths are guaranteed to be at most polynomial in number. Finally, we assert that this new machine in fact is a Few machine that accepts L, by constructing a polynomial-time acceptance predicate Q that decodes the information about the queries to M encoded by our new machine's paths and then uses the acceptance predicate of M and the bounded truth-table reducer to correctly accept or reject.

Proof of Theorem 3.2 It suffices to show that $\mathrm{Few} \supseteq \bigcup_k \mathrm{P}^{\mathrm{Few}[k]}$. Given a set L that (for some constant m) m-tt reduces to a set $M \in \mathrm{Few}$, we wish to show that $L \in \mathrm{Few}$.

The bounded truth-table reduction starts by translating an input into m queries to M in polynomial time. Each query represents a nondeterministic Turing machine computation *that has a polynomially bounded number of accepting paths.* Papadimitriou and Zachos [PZ83] showed that we can encode a polynomial list of NP questions into a single question to a #P (counting) oracle. Cai and Hemachandra [CH86] noted that their technique allows you to recover the number of accepting paths of the original NP questions. Intuitively, one duplicates computation paths to insure that the lowest order bits answer the first question, the next lowest order bits answer the second question, etc.

Using these techniques [PZ83,CH86], the m queries to a Few set could be turned into a single question about the number of accepting paths of a new nondeterministic computation whose number of accepting paths encodes the number of accepting paths of each of the m

queries. However, this new nondeterministic computation might have many solutions. We now describe an improvement on the methods of [PZ83,CH86] that, taking advantage of the fact that the underlying m queries have few accepting paths, allows us to combine them into a single computation tree that has few paths—and that implicitly describes the number of accepting paths of each of the m queries.

Fix a nondeterministic machine $\widehat{M}$ for M. Let q_1, q_2, ..., q_m be the m queries to $M \in$ Few. W.l.o.g., each q_i has at most $|q_i|^l$ accepting paths on this machine. Let $p(\cdot)$ be the polynomial bound on the run time of the btt reduction of L to M. Let us construct a new machine $\widehat{N}$, which on input x constructs the m queries to M and nondeterministically for each i, $1 \leq i \leq m$, does:

1. Clone the current state $2^{(i-1)(1+\lceil \log p(|x|)^l \rceil)}$ times.

2. Run $\widehat{M}(q_i)$.

Let $||N(x)||$ denote the number of accepting paths of N on input x. Note that the lowest order $1 + \lceil \log p(|x|)^l \rceil$ bits of $||\widehat{N}(x)||$ contain $||\widehat{M}(q_1)||$. The next block of $1 + \lceil \log p(|x|)^l \rceil$ bits of $||\widehat{N}(x)||$ contains $||\widehat{M}(q_2)||$, and so on.

So $||\widehat{N}(x)||$ encodes the number of solutions to each of the m queries. Furthermore, on any x, $||\widehat{N}(x)||$ is less than $2^{m(1+\lceil \log p(|x|)^l \rceil)} \leq 2^{2m} p(|x|)^l$ (recall that m is fixed). Thus, $\widehat{N}$ is in fact a machine that never has more than polynomially many paths.

We now argue that $L \in$ Few, by showing that there is a polynomial-time computable predicate Q that causes machine $\widehat{N}$ to accept L in the sense of Definition 1.2. $Q(x, k)$ does the following. $Q(x, k)$ decodes the set of answers to the constituent queries—$q_1, \ldots, q_m$—that k indicates. For each of the m constituent queries, Q use Q_M to determine whether M (the underlying Few set we are reducing to) accepts when we have that number of paths. Since $L \leq^p_{m-tt} M$, $Q(x, k)$ can simluate the polynomial-time truth-table evaluator using the answers Q now knows to the m queries to M. Thus, $Q(x, k)$ can choose to accept if and only if the evaluator says to accept. □

4 Conclusions and Open Problems

This paper has shown that the parity acceptance mechanism is flexible enough to accept any language in Few. In particular, $\oplus$P contains the polynomial accepting path versions of NP, of the counting hierarchy, and of $\mathrm{Mod}_m\mathrm{NP}$, $m > 1$.

We also noted that Few is closed under bounded truth-table reductions.

It is an open question whether Few has complete languages. We suspect that Few may lack complete languages for essentially the same reasons that UP may lack complete languages [HH86]—enumeration of path-restricted machines is troublesome.

The central open question remaining is whether $\oplus P \supseteq NP$, or $NP \supseteq \oplus P$, or neither. We conjecture that $\oplus P$ and NP are incomparable ($\oplus P \not\subseteq NP$ and $NP \not\subseteq \oplus P$), and thus that our inclusion $\oplus P \supseteq FewNP$ will not be strengthened to $\oplus P \supseteq NP$.

However, even to display an oracle A for which $\oplus P^A \not\supseteq NP^A$ seems difficult. By the result of this paper, such an oracle would have to create a language in NP^A that is complex enough not to be in $FewNP^A$. It is possible, though, to construct an oracle A for which $\oplus P^A \not\subseteq NP^A$ (either by direct diagonalization, or as a corollary of the results of [Yao85] and [Cai86]).

Acknowledgements

We thank Eric Allender, David Eppstein, Judy Goldsmith, and Juris Hartmanis for helpful conversations, references, and advice.

References

[All85] E. Allender. Invertible functions. 1985. Ph.D. thesis, Georgia Institute of Technology.

[All86] E. Allender. The complexity of sparse sets in P. In *Proceedings 1st Structure in Complexity Theory Conference*, pages 1–11, Springer-Verlag *Lecture Notes in Computer Science #223*, June 1986.

[BG81] C. Bennett and J. Gill. Relative to a random oracle A, $P^A \neq NP^A$ with probability 1. *SIAM J. on Computing*, 10:96–113, 1981.

[BG82] A. Blass and Y. Gurevich. On the unique satisfiability problem. *Information and Control*, 55:80–88, 1982.

[Cai86] J. Cai. With probability one, a random oracle separates PSPACE from the polynomial-time hierarchy. In *18th ACM Symposium on Theory of Computing*, pages 21–29, 1986.

[CGH*86] J. Cai, T. Gundermann, J. Hartmanis, L. Hemachandra, V. Sewelson, K. Wagner, and G. Wechsung. The boolean hierarchy II: applications. 1986. Submitted.

[CH86] J. Cai and L. Hemachandra. *Exact Counting is as Easy as Approximate Counting.* Technical Report TR86-761, Cornell Department of Computer Science, Ithaca, NY, June 1986.

[GHJY87] J. Goldsmith, L. Hemachandra, D. Joseph, and P. Young. Near-testable sets. 1987. In preparation.

[GJY87] J. Goldsmith, D. Joseph, and P. Young. Self-reducible, P-selective, near-testable, and P-cheatable sets: the effect of internal structure on the complexity of a set. In *Proceedings 2nd Structure in Complexity Theory Conference*, pages 50–59, 1987.

[GS84] J. Grollmann and A. Selman. Complexity measures for public-key cryptosystems. In *Proceedings 25th IEEE Symposium on Foundations of Computer Science*, pages 495–503, 1984.

[GW87] T. Gundermann and G. Wechsung. Counting classes with finite acceptance types. 1987. To appear.

[Hem87a] L. Hemachandra. *On Parity and Near-Testability:* $P^A \neq NT^A$ *With Probability 1.* Technical Report 87-852, Cornell University, Department of Computer Science, Ithaca, NY, July 1987.

[Hem87b] L. Hemachandra. The strong exponential hierarchy collapses. In *19th ACM Symposium on Theory of Computing*, pages 110–122, May 1987.

[HH86] J. Hartmanis and L. Hemachandra. Complexity classes without machines: On complete languages for UP. In *Automata, Languages, and Programming (ICALP 1986)*, pages 123–135, Springer-Verlag *Lecture Notes in Computer Science #226*, July 1986.

[HH87] J. Hartmanis and L. Hemachandra. One-way functions, robustness, and the non-isomorphism of NP-complete sets. In *Proceedings 2nd Structure in Complexity Theory Conference*, pages 160–174, IEEE Computer Society Press, June 1987.

[JY85] D. Joseph and P. Young. Some remarks on witness functions for non-polynomial and non-complete sets in NP. *Theoretical Computer Science*, 39:225–237, 1985.

[Lau83] C. Lautemann. BPP and the polynomial hierarchy. *Information Processing Letters*, 14:215–217, 1983.

[LLS75] R. Ladner, N. Lynch, and A. Selman. A comparison of polynomial time reducibilities. *Theoretical Computer Science*, 1(2):103–124, 1975.

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

[PZ83] C. Papadimitriou and S. Zachos. Two remarks on the power of counting. In *Proceedings 6th GI Conference on Theoretical Computer Science*, pages 269–276, Springer-Verlag Lecture Notes in Computer Science #145, 1983.

[Sch87] U. Schöning. Probabilistic complexity classes and lowness. In *Proceedings 2nd Structure in Complexity Theory Conference*, pages 2–8, IEEE Computer Society Press, June 1987.

[Sim77] J. Simon. On the difference between one and many. In *Automata, Languages, and Programming (ICALP 1977)*, pages 480–491, Springer-Verlag *Lecture Notes in Computer Science #52*, 1977.

[Ukk83] E. Ukkonen. Two results on polynomial time truth-table reductions to sparse sets. *SIAM Journal on Computing*, 12(3):580–587, 1983.

[Val76] L. Valiant. The relative complexity of checking and evaluating. *Information Processing Letters*, 5:20–23, 1976.

[Wec85] G. Wechsung. *More about the Closure of NP.* Technical Report TR N/85/43, Friedrich-Schiller-Universität, December 1985.

[Yao85] A. Yao. Separating the polynomial-time hierarchy by oracles. In *Proceedings 26th IEEE Symposium on Foundations of Computer Science*, pages 1–10, 1985.

[Yes83] Y. Yesha. On certain polynomial-time truth-table reducibilities of complete sets to sparse sets. *SIAM Journal on Computing*, 12(3):411–425, 1983.

[Zac86] S. Zachos. Probabilistic quantifiers, adversaries, and complexity classes: an overview. In *Proceedings 1st Structure in Complexity Theory Conference*, pages 383–400, IEEE Computer Society Press, June 1986.