On Truth-Table Reducibility to SAT

Samuel R. Buss[*]
Department of Mathematics
University of California, San Diego

Louise Hay[†]
Department of Mathematics, Statistics, and Computer Science
University of Illinois at Chicago

July 11, 2002

Abstract

We show that polynomial time truth-table reducibility via Boolean circuits to SAT is the same as logspace truth-table reducibility via Boolean formulas to SAT and the same as logspace Turing reducibility to SAT. In addition, we prove that a constant number of rounds of parallel queries to SAT is equivalent to one round of parallel queries. We give an oracle relative to which Δ_2^p is not equal to the class of predicates polynomial time truth-table reducible to SAT.

1 Introduction

Ladner, Lynch, and Selman [17] introduced the notion of polynomial time truth-table reducibility as an aid to the study of low level complexity. As noted by Ladner and Lynch [16] this is equivalent to the notion of polynomial

[*]Supported in part by NSF grants DMS-8511465, DMS-8701828 and DMS-8902480.
[†]Work partially completed while on sabbatical leave at the University of California, Berkeley

time truth-table reducibility via Boolean circuits; they asked whether it is equivalent to polynomial time truth-table reducibility via Boolean formulas. This paper studies the class of predicates which are polynomial time truth-table reducible to SAT; a number of equivalent and somewhat surprising equivalent definitions are given. In particular, we show that, for reducibility to the NP-complete problem SAT, polynomial time truth-table reducibility via Boolean circuits is equivalent to logspace truth-table reducibility via Boolean formulas. Consequently, it is also equivalent to logspace Turing reducibility to SAT. Several other equivalent definitions are given for the class of predicates polynomial time truth-table reducible to SAT and the fact that all these definitions coincide provide evidence that this is a robust and interesting class for computational complexity. The results below refute a conjecture of Köbler, Schöning and Wagner [13], but answer only a special case of the original questions of Ladner and Lynch.

Let $\leq_{tt}^{p}(\mathrm{NP})$ denote the class of decision problems polynomial time truth-table reducible to NP; this paper presents several other characterizations of this class. First, Krentel [14] defines the class $\mathrm{P}^{\mathrm{SAT}}[O(\log n)]$; we show this is equal to $\leq_{tt}^{p}(\mathrm{NP})$. Cai and Hemachandra [7] defined a hierarchy Σ_k^L, Π_k^L above the finite levels of the difference hierarchy over NP; we show this hierarchy collapses with $\Sigma_2^L = \Pi_2^L = \leq_{tt}^{p}(\mathrm{NP})$. Furthermore, $\leq_{tt}^{p}(\mathrm{NP})$ is precisely the set of predicates definable by $\Sigma_2^b \cap \Pi_2^b$ formulas as defined by Buss [2].

The class $\leq_{tt}^{p}(\mathrm{NP})$ is defined so that it contains the predicates which are computed in polynomial time by a Turing machine which is allowed one set of parallel queries to SAT. This paper shows that in fact a constant number of rounds of parallel queries to SAT are permissable — so a polynomial time Turing machine which makes $O(1)$ rounds of parallel queries to SAT recognizes a $\leq_{tt}^{p}(\mathrm{NP})$ predicate.

In the final section we provide evidence that $\leq_{tt}^{p}(\mathrm{NP})$ is distinct from other commonly studied classes in compuational complexity by giving one oracle relative to which $\leq_{tt}^{p}(\mathrm{NP})$ is equal to PSPACE and another oracle relative to which $\leq_{tt}^{p}(\mathrm{NP})$ and PSPACE are distinct. For both oracles, the Boolean hierarchy over NP is proper and hence distinct from $\leq_{tt}^{p}(\mathrm{NP})$ and, in particular, $\leq_{tt}^{p}(\mathrm{NP})$ is not equal to NP relative to these oracles.

Many of the results of this paper have been obtained independently by K. Wagner; in particular, Theorem 1 and many of the other results of section 2 are in Wagner [21, 20]. In addition, one of the referees pointed out that alternative proofs of most of the results in sections 1 through 3 can be obtained using the 'census function' method of Hemachandra [11]. Similar 'counting' techniques have been used recently by a number of authors, an early example is Jenner, Kirsig and Lange [12]; the work of Beigel [1] was particularly helpful to us in the original formulation of our results.

The current paper is a slightly revised version of part of [4]; we wish to thank the referees for a number of suggestions which improved this paper.

2 Truth-Table Reducibility to SAT

Polynomial time truth-table reducibility ($\leq_{tt}^{p}$) was first introduced by Ladner, Lynch and Selman [17] and further studied by Ladner and Lynch [16]. The general idea is that $A \leq_{tt}^{p} B$ if there is a polynomial time function which, on input x, precomputes a number of questions about membership in B and describes a Boolean function of the answers to these questions which specifies the membership of x in A. However, in the low level complexity setting there are several, apparently inequivalent ways of describing a Boolean function; the most common representations are as a Boolean *circuit*, a Boolean *formula* or as a full truth table. These representations may give rise to different notions of polynomial time truth-table reducibilities since there may be no way to convert one representation to another in polynomial time; indeed, it is an open question whether Boolean circuits have equivalent polynomial size Boolean formulas. And of course, a full truth-table may be exponentially larger than an equivalent Boolean formula. Accordingly, we shall define three different versions of polynomial time truth-table reducibilities. (These have already been considered by Ladner and Lynch and more recently by Wagner [19] and Köbler, Schöning and Wagner [13].)

2.1 Some Definitions and a Main Theorem

As usual, P and NP denote the class of polynomial time and nondeterministic polynomial time predicates, respectively. Let $\leq_{T}^{p}$ denote polynomial time, Turing reduction (as compared to many-one reduction). Likewise, $\leq_{T}^{log}$ denotes logspace, Turing reduction.

Definition Let A and B be decision problems. Let $\mathrm{eval}(z, x_1, \ldots, x_n)$ be the value of the Boolean function coded by z on binary inputs $x_1, \ldots, x_n$. Let

χ_B be the characteristic function of B, so $\chi_B(y) = 1$ if $y \in B$ and $\chi_B(y) = 0$ otherwise. Then:

(1) $A \leq_{tt}^{p} B$, A is *polynomial time (Boolean circuit) truth-table reducible to* B if and only if there is a polynomial time function f such that for all x, $f(x) = \langle z, y_1, \ldots, y_k \rangle$ with z coding a Boolean circuit such that

$$x \in A \iff \text{eval}(z, \chi_B(y_1), \ldots, \chi_B(y_k)) = 1.$$

(2) $A \leq_{bf}^{p} B$, A is *polynomial time, Boolean formula truth-table reducible to* B if and only if there is a polynomial time function f such that for all x, $f(x) = \langle z, y_1, \ldots, y_k \rangle$ with z coding a Boolean formula such that

$$x \in A \iff \text{eval}(z, \chi_B(y_1), \ldots, \chi_B(y_k)) = 1.$$

(3) $A \leq_{ftt}^{p} B$, A is *polynomial time, full truth-table reducible to* B if and only if there is a polynomial time function f such that for all x, $f(x) = \langle z, y_1, \ldots, y_k \rangle$ with z coding a full truth-table such that

$$x \in A \iff \text{eval}(z, \chi_B(y_1), \ldots, \chi_B(y_k)) = 1.$$

In all three cases, the integer k may depend on x.

Ladner, Lynch and Selman showed that $A \leq_{tt}^{p} B$ if and only if A is polynomial time reducible to B with one round of parallel queries (see section 3).

Definition Let A be a decision problem. Then $\leq_{tt}^{p}(A)$, $\leq_{bf}^{p}(A)$ and $\leq_{ftt}^{p}(A)$ are the classes of decision problems which are polynomial time truth-table reducible to A via Boolean circuits, via Boolean formulas and via full truth tables, respectively. Also, $\leq_{tt}^{p}(\text{NP})$, $\leq_{bf}^{p}(\text{NP})$ and $\leq_{ftt}^{p}(\text{NP})$ denote $\leq_{tt}^{p}(\text{SAT})$, $\leq_{bf}^{p}(\text{SAT})$ and $\leq_{ftt}^{p}(\text{SAT})$, respectively.

The definitions of polynomial time truth-table reductions may be adapted for logspace computability as well; except that for logspace reductions there is a fourth natural notion of truth-table reducibility. We define $\leq_{tt}^{log}$, $\leq_{bf}^{log}$ and $\leq_{ftt}^{log}$ exactly as in the definitions above except that now the function f is required to be logspace computable instead of polynomial time computable. Actually, the reducibility $\leq_{tt}^{log}$ is somewhat counterintuitive since the evaluation of Boolean circuits is not logspace computable unless P equals logspace.

For this reason, Ladner and Lynch considered a form of logspace reducibility which we shall denote $\leq_{LL}^{log}$ which is intermediate in strength between $\leq_{bf}^{log}$ and $\leq_{tt}^{log}$. Referring to the definitions above, Ladner and Lynch called our z coding a Boolean function a 'tt-condition' and called the *eval* function a 'tt-condition evaluator'. For $\leq_{LL}^{log}$ reducibility there is no restriction on the format of the code z but *eval* is required to be logspace computable. By the theorem of Lynch [18] that Boolean formulas can be evaluated in logspace, if $A \leq_{bf}^{log} B$ then $A \leq_{LL}^{log} B$; and by Ladner's theorem [15] that the circuit value problem is complete for P, if $A \leq_{LL}^{log} B$ then $A \leq_{tt}^{log} B$.

It is obvious that $\leq_{tt}^{p}(\mathrm{NP}) \supseteq \leq_{bf}^{p}(\mathrm{NP}) \supseteq \leq_{ftt}^{p}(\mathrm{NP})$; complete problems for these classes are given by Köbler, Schöning and Wagner [13] and in Wagner [19]. The next theorem states that $\leq_{tt}^{p}(\mathrm{NP})$ is actually equal to $\leq_{bf}^{p}(\mathrm{NP})$ and the following theorem and corollary show that they are equal to all of $\leq_{bf}^{log}(\mathrm{NP})$, $\leq_{LL}^{log}(\mathrm{NP})$, $\leq_{tt}^{log}(\mathrm{NP})$ and $\leq_{T}^{log}(\mathrm{NP})$. This theorem was independently discovered by K. Wagner.

Theorem 1 $\leq_{tt}^{p}(\mathrm{NP}) = \leq_{bf}^{p}(\mathrm{NP})$.

Proof Let Σ be a finite alphabet and suppose $C \subseteq \Sigma^*$ and $C \leq_{tt}^{p} \mathrm{SAT}$; let f be a polynomial time function giving a truth-table reduction of C to SAT. For all $x \in \Sigma^*$, $f(x)$ codes a sequence $\langle c, f_1(x), \ldots, f_k(x)\rangle$; we shall sometimes write $c(x)$ and $k(x)$ to make explicit that c and k are (polynomial time) functions of x. Since f is polynomial time computable, $k(x)$ must be bounded by a polynomial $q(n)$ where $n = |x|$ is the length of x. Now $\mathrm{SAT}(y)$ is the predicate "y is satisfiable", also let $\mathrm{TRU}(u, y)$ be the polynomial time predicate "u codes a satisfying assignment of y"; so $\mathrm{SAT}(y) \leftrightarrow (\exists u < y)\mathrm{TRU}(u, y)$. Let $B(i, x)$ be the predicate

$$(\exists w = \langle w_1, \ldots, w_{k(x)}\rangle)\,[\text{for } i \text{ distinct values of } j,\ \mathrm{TRU}(w_j, f_j(x))]$$

which asserts that there are satisfying assignments for $\geq i$ of $f_1(x), \ldots, f_k(x)$. Clearly $B \in \mathrm{NP}$. Let $A(i, x)$ be the predicate

$$\begin{aligned}(\exists w = \langle w_1, \ldots, w_{k(x)}\rangle)[&\text{for } i \text{ distinct values of } j,\ \mathrm{TRU}(w_j, f_j(x)), \text{ and}\\ &\mathrm{eval}(c(x), \chi_{\mathrm{TRU}}(w_1, f_1(x)), \ldots, \chi_{\mathrm{TRU}}(w_k, f_k(x))) = 1\,]\end{aligned}$$

So $A(i, x)$ is in NP and asserts that there are satisfying assignments for i of the $f_j(x)$'s such that evaluating the Boolean circuit $c(x)$ assuming that these are all the satisfying assignments yields a *True* result.

Note that if $A(i,x)$ is true then $B(j,x)$ is true for all $j \leq i$. Also, if $B(i,x)$ is true and $B(i+1,x)$ is false then there are exactly i distinct values of j for which $f_j(x) \in \mathrm{SAT}$. Hence C can be defined by

$$x \in C \iff (\exists i \leq k(x))(A(i,x) \wedge \neg B(i+1,x)).$$

Finally we claim that this puts C in $\leq_{bf}^{p}(\mathrm{SAT})$. This is because for any x, $x \in C$ if and only if the Boolean formula

$$\bigvee_{i=0}^{k(x)} (A(i,x) \wedge \neg B(i+1,x))$$

evaluates to be *True*. Since SAT is NP-complete, there are polynomial time functions f_A and f_B so that $f_A(i,x) \in \mathrm{SAT}$ iff $A(i,x)$ and similarly for B. Hence for any x, $x \in C$ if and only if the Boolean formula

$$\bigvee_{i=0}^{k(x)} (\chi_{\mathrm{SAT}}(f_A(i,x)) \wedge \neg\chi_{\mathrm{SAT}}(f_B(i+1,x))).$$

evaluates to *True*. Since $k(x)$ is bounded by a polynomial of the length of x, this is clearly a polynomial time, Boolean formula, truth-table reduction of C to SAT.

Q.E.D. Theorem 1 $\square$

Actually the above proof gives a stronger result:

Theorem 2 *A decision problem C is in $\leq_{tt}^{p}(\mathrm{NP})$ if and only if it is logspace truth-table reducible to* SAT *via Boolean formulas. So $\leq_{tt}^{p}(\mathrm{NP})$ equals $\leq_{bf}^{log}(\mathrm{NP})$*

Proof The above proof still applies; note that f_A and f_B can be picked to be logspace computable. $\square$

Corollary 3 *A decision problem C is in $\leq_{tt}^{p}(\mathrm{NP})$ if and only if it is logspace Turing reducible to* SAT.

Proof This is immediate from Theorem 2 and the theorem of Ladner and Lynch [16] that logspace Turing reducibility is identical to the logspace truth-table reducibility $\leq_{LL}^{log}$. In general, $\leq_{bf}^{log}$ is weaker than $\leq_{LL}^{log}$, the latter is weaker than $\leq_{tt}^{log}$, which in turn is weaker than $\leq_{tt}^{p}$. It follows from Theorem 2 that all these reducibilities are equivalent for reductions to SAT. $\square$

Theorem 4 *Every predicate in $\leq_{tt}^p(\mathrm{NP})$ can be defined by a formula of the form*

$$(\exists i \leq p(|x|))[A(i,x) \wedge \neg B(i,x)]$$

and by a formula of the form

$$(\forall i \leq q(|x|))[C(i,x) \vee \neg D(i,x)]$$

where A, B, C and D are NP*-predicates and p and q are polynomials.*

Proof The proof of Theorem 1 proves the first part; the second part is a consequence of the fact that $\leq_{tt}^p(\mathrm{NP})$ is closed under complementation. $\square$

2.2 A Syntactic Characterization of $\leq_{tt}^p(\mathrm{NP})$

In this and the next subsection, we give alternate characterizations of the the class $\leq_{tt}^p(\mathrm{NP})$; these follow as easy corollaries of the above theorems.

First we provide a syntactic characterization of formulas which define predicates in $\leq_{tt}^p(\mathrm{NP})$ by showing that a decision problem is in $\leq_{tt}^p(\mathrm{NP})$ if and only if it is definable by a $\Sigma_2^b \cap \Pi_2^b$-formula in the sense of [2]. The superscript b is used to indicate that Σ_2^b and Π_2^b are classes of *formulas*; the Σ_2^b (respectively, the Π_2^b) formulas define precisely the Σ_2^p (respectively, the Π_2^p) predicates. Hence we shall establish that a decision problem is in $\leq_{tt}^p(\mathrm{NP})$ if and only if it has a defining formula which is simultaneously and explicitly in both Σ_2^b and Π_2^b. (However, this is in no way intended to say that $\Sigma_2^p \cap \Pi_2^p = \leq_{tt}^p(\mathrm{NP})$.)

Buss [2] defines a first-order language for bounded arithmetic where variables ranges over integers and with non-logical symbols 0, S, $+$, $\cdot$, $|x|$, $\lfloor \frac{1}{2}x \rfloor$, $\#$ and $\leq$ where $|x|$ equals the length of the binary representation of x and $x\#y = 2^{|x|\cdot|y|}$. A *bounded quantifier* is a quantifier of the form $(Qx \leq t)$ where Q_i is $\forall$ or $\exists$ and t is a term; a *sharply bounded quantifier* is a bounded quantifier with the bound t of the form $t = |s|$ for some term s. A *bounded formula* is a formula in which all quantifiers are bounded. The classes Σ_i^b and Π_i^b are classes of bounded formulas defined by counting alternations of (bounded) quantifiers, ignoring the sharply bounded ones. The formulas in the class Σ_i^b or Π_i^b define precisely the predicates in the i-th level of the polynomial time hierarchy Σ_i^p or Π_i^b, respectively. The class $\Sigma_2^b \cap \Pi_2^b$ has already been studied by Buss [3] who showed that it is the largest class of formulas for which S^1_2 is known to prove PIND (polynomial- or length-induction).

For the reader who has not studied bounded arithmetic, we provide some alternate, essentially equivalent definitions:

Alternate Definitions: A *sharply bounded quantifier* is a quantifier of the form $(\forall y \leq p(|\vec{x}|))$ or $(\exists y \leq p(|\vec{x}|))$ for p a polynomial and $|\vec{x}|$ the the total length of the binary representations of $\vec{x}$. An NP-*formula* is a formula which defines a predicate in NP. $\Sigma^b_2 \cap \Pi^b_2$ is the smallest class of formulas which contains all NP-formulas, is closed under the Boolean operations of negation $(\neg)$, conjunction $(\wedge)$ and disjunction $(\vee)$ and is closed under sharply bounded quantification.

Theorem 5 *The class* $\leq^p_{tt}(\mathrm{NP})$ *is equal to the class of predicates definable by* $\Sigma^b_2 \cap \Pi^b_2$ *formulas.*

Proof Theorem 4 states that every predicate in $\leq^p_{tt}(\mathrm{NP})$ can be expressed by a formula of the form

$$(\exists i \leq p(|x|))\,[A(i,x) \wedge \neg B(i+1,x)]$$

where A and B define NP predicates. This is clearly a $\Sigma^b_2 \cap \Pi^b_2$ formula. For the converse, it is easy to prove by induction on the complexity of formulas that every $\Sigma^b_2 \cap \Pi^b_2$ formula $C(x)$ is equivalent to a formula of the form

$$(Q_1 y_1 \leq p(|x|))(Q_2 y_2 \leq p(|x|)) \cdots (Q_k y_k \leq p(|x|))\Psi$$

where p is a polynomial and Ψ is a Boolean combination of NP formulas. Since SAT is NP-complete we may assume that Ψ is a Boolean combination of formulas of the form $\mathrm{SAT}(f(x,\vec{y}))$ with f a polynomial time computable function. To give a truth-table reduction of $\{x : C(x)\}$ to SAT we just express $C(x)$ as a Boolean formula

$$\mathop{\bigvee\!\!\!\!\!\!\bigwedge}_{y_1=0}^{p(|x|)} \mathop{\bigvee\!\!\!\!\!\!\bigwedge}_{y_2=0}^{p(|x|)} \cdots \mathop{\bigvee\!\!\!\!\!\!\bigwedge}_{y_k=0}^{p(|x|)} \Psi$$

where $\bigvee\!\!\!\!\!\!\bigwedge$ denotes a $\bigvee$ or a $\bigwedge$ depending on whether Q_i is $\exists$ or $\forall$. Clearly this Boolean formula is obtainable as a function of x in polynomial time. $\square$

Cai and Hemachandra [7] define a hierarchy of sets denoted Σ_k^L, Π_k^L for $k \geq 0$. They defined these sets in terms of Turing machine computation: a predicate is in Σ_k^L (or Π_k^L) if it is recognized by a Turing machine which has k logtime alternation blocks beginning with an existential (universal, respectively) block and followed by a polynomial time alternation block which may be existential on some inputs and universal on other inputs. It is readily seen that the following syntactic definition is equivalent:

Definition Let Σ be an alphabet. A predicate $A \subseteq \Sigma^*$ is in Σ_k^L if and only if A can be defined as the set of $x \in \Sigma^*$ such that

$$\begin{aligned}&(\exists y_1 \leq p_1(|x|))(\forall y_2 \leq p_2(|x|)) \cdots (Q_k y_k \leq p_k(|x|))\\ &\qquad [(R(x,\vec{y}) \to B(x,\vec{y})) \wedge (\neg R(x,\vec{y}) \to C(x,\vec{y}))]\end{aligned}$$

where Q_k is $\exists$ or $\forall$ depending on whether k is odd or even, R is a polynomial time predicate, B is an NP predicate and C is a co-NP predicate. A is a Π_k^L predicate if and only if the complement of A is a Σ_k^L predicate.

Cai and Hemachandra left open the question of whether this hierarchy is proper; in fact, it collapses by the second level:

Theorem 6 $\Sigma_k^L = \Pi_k^L = \Sigma_2^L = \Pi_2^L = \leq_{tt}^p(\mathrm{NP})$ *for all* $k \geq 2$.

Proof Clearly Σ_k^L and Π_k^L have $\Sigma_2^b \cap \Pi_2^b$ definitions. Hence Σ_k^L and Π_k^L are subsets of $\leq_{tt}^p(\mathrm{NP})$. Furthermore, we claim that any $C \in \leq_{tt}^p(\mathrm{NP})$ is in Σ_2^L. By Theorem 4, C is defined by a formula

$$(\exists i \leq p(|x|))(A(i,x) \wedge \neg B(i,x))$$

with A, $B \in \mathrm{NP}$. So C can be defined by

$$(\exists i \leq p(|x|))(\forall j \leq 1)[(j = 0 \to A(i,x)) \wedge (j \neq 0 \to \neg B(i,x))].$$

Hence C is in Σ_2^L. Since $\leq_{tt}^p(\mathrm{NP})$ is closed under complementation, $\leq_{tt}^p(\mathrm{NP})$ is also a subset of Π_2^L and we are done. $\Box$

2.3 Another Characterization of $\leq_{tt}^p(\mathrm{NP})$

Krentel [14] defines $\mathrm{P}^{\mathrm{SAT}}[O(\log n)]$ to be the set of predicates recognizable in polynomial time with an oracle for SAT with the restriction that SAT may be queried only $O(\log n)$ times on inputs of length n.

Theorem 7 $\mathrm{P}^{\mathrm{SAT}}[O(\log n)] = \leq_{tt}^{p}(\mathrm{NP})$.

This theorem is well-known but we include its proof for the sake of completeness. Hemachandra [11] has already established the harder ($\supseteq$) direction; we first found the proof using ideas from Beigel [1]. The method of proof is also used in section 4.

Proof $\subseteq$: Suppose $A \in \mathrm{P}^{\mathrm{SAT}}[O(\log n)]$ and A is recognized by Turing machine M^{SAT} in polynomial time with only $O(\log n)$ queries. Because of the restriction on the number of queries there are only polynomially many possible queries for a given input x to M (regardless of what the answers to the queries may be). This is because there are only polynomially many possible answers of $O(\log n)$ queries during the execution of $M^{\Omega}(x)$ for an arbitrary oracle Ω. This makes it possible to give a polynomial time truth-table reduction of A to SAT: given x, compute all potential queries of $M^{\Omega}(x)$, then write out a Boolean circuit which gives the value $M^{\mathrm{SAT}}(x)$ in terms of the answers to the queries to SAT.

$\supseteq$: Let C be in $\leq_{tt}^{p}(\mathrm{NP})$ and let $A(i,x)$ and $B(i,x)$ be as in the proof of Theorem 1, so that, for all x,

$$x \in C \iff (\exists i \leq k(x))[A(i,x) \wedge \neg B(i+1,x)].$$

Let $A^*(i,x)$ be $A(i,x) \vee B(i+1,x)$; clearly A^* is an NP predicate and, by the definitions of $A(i,x)$ and $B(i,x)$,

$$x \in C \iff (\exists i \leq k(x))[A^*(i,x) \wedge \neg B(i+1,x)].$$

Now note that for all i, $B(i+1,x)$ implies $A^*(i,x)$ and $A^*(i,x)$ implies $B(i,x)$. So consider the sequence of NP predicates

$$A^*(0,x), B(1,x), A^*(1,x), B(2,x), \ldots, A^*(k(x),x), B(k(x)+1,x)$$

where each predicate is implied by the next. Of course $B(k(x)+1,x)$ must be false; if we knew which NP predicate in the sequence is the *first* false one, then we would know if $x \in C$. This is because if $A^*(i,x)$ is true and $B(i+1,x)$ is false then $x \in C$ whereas if $B(i,x)$ is true and $A^*(i,x)$ is false then $x \notin C$. But it is easy to see that a binary search can be used to find the first false predicate in polynomial time with $O(\log n)$ queries to SAT. $\Box$

The above method of proof is essentially equivalent to, although cruder than, Beigel's proof that k parallel queries to SAT are polynomial time equivalent to 2^k serial queries to SAT; his proof used "mind-changes". The crucial idea here is the use of a descending sequence of NP predicates; this idea has been exploited to give a definition of an infinite difference hierarchy over NP in [4].

Definition Let A and B be predicates. A function f is a *normalized* truth-table reduction of A to B if and only if for all x, $f(x) = \langle y_1, \ldots, y_{k(x)} \rangle$ so that $\chi_B(y_i) \geq \chi_B(y_{i+1})$ for all i and so that $x \in A$ if and only if an odd number of the y_i's are in B.

Theorem 8 *A predicate A is in $\leq^p_{tt}(\mathrm{NP})$ if and only if there is a polynomial time, normalized truth-table reduction of A to* SAT*.*

The theorem is immediate from the proofs of Theorems 1 and 7. The fact that 'odd' rather than 'even' is used in the definition of normalized truth-table reductions is unimportant since f could be modified to prepend a $y_0 \in B$ to each sequence. Normalized truth-table reductions have already been studied by Wagner [19] under the name 'truth-table Hausdorf reductions' and Theorem 8 is also a direct consequence of Theorem 2 and a theorem of Wagner's that $\leq^p_{bf}(NP)$ equals the set of predidates which are polynomial time, truth-table Hausdorf reducible to an NP set.

3 Multiple Rounds of Parallel Queries

Ladner, Lynch and Selman [17] proved that $\leq^p_{tt}(\mathrm{NP})$ contains precisely those decision problems $A(x)$ which are recognized by a polynomial time Turing machine which is allowed one round of parallel queries to SAT. By a *round of parallel queries to* SAT we mean that the Turing machine writes a set of strings separated by delimiters on a query tape and then invokes an oracle for SAT; the oracle returns a string of Yes/No answers on an answer tape which specify membership of each query string in SAT.

A crucial feature of the notion of allowing one round of parallel queries to SAT is that all the queries must be formulated before any answers are known. This leads naturally to the question of whether two (say) rounds of queries to SAT are more powerful than one. For instance, in two rounds the Turing machine could ask polynomially many queries to SAT in parallel and then use the answers to construct a further query to SAT. One's initial impression is that this may provide more computational power; however, we prove below the somewhat surprising result that, for polynomial time Turing machines, a constant number of rounds of parallel queries to SAT can be reduced to one round of parallel queries.

Theorem 9 *Let $k \geq 1$. If $A(x)$ is a decision problem recognized by a Turing machine M which runs in polynomial time and makes k rounds of parallel queries to* SAT*, then $A(x)$ is in $\leq_{tt}^{p}(\mathrm{NP})$, i.e., $A(x)$ is polynomial time truth-table reducible to* SAT.

The idea of the proof of Theorem 9 is very similar to the proof of Theorem 1.

Proof Fix k, an integer ≥ 1. Let us assume that $M(x)$ always asks $p(n)$ or fewer queries in a given round, where $n = |x|$ and p is a polynomial. Recall that $\mathrm{TRU}(x, y)$ is the predicate "x is a satisfying assignment for y". A k-tuple of integers is a sequence $\vec{r} = \langle r_1, \ldots, r_k \rangle$ of integers. We order the k-tuples lexicographically so $\vec{r} \prec \vec{s}$ iff for some i, $r_i < s_i$ and $(\forall j < i)(r_j = s_j)$.

If $\vec{w} = \langle w_1, \ldots, w_\ell \rangle$ is a sequence of truth assignments, we define $M^{\vec{w}}$ to be the polynomial time Turing machine which runs as follows: on input x, $M^{\vec{w}}$ runs like M, except when a query state is entered $M^{\vec{w}}$ does not invoke an oracle for SAT but instead, for each query y, checks if there is some member w_j of $\vec{w}$ such that $\mathrm{TRU}(w_j, y)$ is satisfied. If so, $M^{\vec{w}}$ acts as if the oracle had answered "Yes, $y \in \mathrm{SAT}$"; in this case we say that $M^{\vec{w}}$ *sees y satisfied.* Otherwise, of course, $M^{\vec{w}}$ acts as if the oracle had answered "No, $y \notin \mathrm{SAT}$". Note that $M^{\vec{w}}(x)$ is polynomial time in both x and $\vec{w}$.

Now let $\vec{r}(\vec{w}, x) = \langle r_1(\vec{w}, x), \ldots, r_k(\vec{w}, x) \rangle$ be defined so that $r_j(\vec{w}, x)$ is equal to the number of queries y that $M^{\vec{w}}(x)$ sees satisfied in the j-th round of parallel queries to SAT. In other words, $r_j(\vec{w}, x)$ is equal to the number of queries y asked in the j-th round of $M^{\vec{w}}(x)$ which are satisfied by some member of $\vec{w}$. Clearly $\vec{r}(\vec{w}, x)$ is a polynomial time function of x and $\vec{w}$. We can express membership in C by:

$$x \in C \leftrightarrow \exists \vec{s} = \langle s_1, \ldots, s_k \rangle (\exists \vec{w} [M^{\vec{w}}(x) \text{ accepts} \wedge \vec{s} = \vec{r}(\vec{w}, x)] \wedge \\ \wedge \neg \exists \vec{v} (\vec{s} \prec \vec{r}(\vec{v}, x))).$$

Note that the quantifiers $\exists \vec{w}$ and $\exists \vec{v}$ may be polynomially bounded since $M(x)$ can make only $(p(n))^k$ total queries and hence, for instance, $\vec{w}$ can be assumed to be of the form $\langle w_1, \ldots, w_\ell \rangle$ with $\ell \leq (p(n))^k$ and with each w_j of length less that the total runtime of $M(x)$. Thus the two predicates inside the expression for $x \in C$ of the forms $\exists \vec{w}(\cdots)$ and $\exists \vec{v}(\cdots)$ are in NP. Also note the values s_j may be restricted to be $\leq p(n)$ so the $\exists \vec{s}$ quantifier is sharply bounded. Thus by Theorem 5, C is in $\leq_{tt}^p(\mathrm{NP})$. (We could avoid the use of Theorem 5 by directly expanding the $\exists \vec{s}$ quantifier as a disjunction and thereby express $x \in C$ as a polynomial size Boolean formula involving instances of SAT.) $\Box$

Finally, note that the k in Theorem 9 must be constant. If k is unrestricted we get all of Δ_2^p, the class of predicates polynomial time Turing reducible to NP. However, we can let k vary a little if we bound severely the number of parallel queries to SAT allowed in a round. Indeed, if $M(x)$ makes $f_j(x)$ queries in the j-th round and if, for all x, $\sum \log(f_j(x)+1) = O(\log n)$, or equivalently, if $\prod(f_j(x)+1) = n^{O(1)}$, then M recognizes a predicate in $\leq_{tt}^p(\mathrm{NP})$. This fact has the $\subseteq$-inclusion of Theorem 7 as a special case with $k = O(\log n)$ and $f_j = O(1)$.

4 Separation Results

It is obvious that $\leq_{tt}^p(\mathrm{NP})$ includes NP and is a subset of $\Delta_2^p = \leq_T^p(\mathrm{NP})$; it is of course open whether these three classes are distinct since, if so, then $\mathrm{P} \neq \mathrm{NP}$. It is thus natural to ask whether these classes are distinct in relativized worlds.

There are some related prior results for the Boolean (or difference) hierarchy over NP. The difference hierarchy was defined in an abstract setting by Hausdorff [10] and the Boolean hierarchy over NP has been extensively studied by a number of researchers [22, 8, 13, 5, 6]. Cai and Hemachandra [8] showed that there are oracles relative to which the Boolean hierarchy over has infinitely many levels, and oracles relative to which the Boolean hierarchy extends exactly k levels, with PSPACE collapsing to the $(k+1)$-st level. We extend these results by showing that in relativized worlds where the Boolean hierarchy over NP has infinitely many levels, the condition of $\leq_{tt}^p(\mathrm{NP}^A)$ being equal to $\leq_T^p(\mathrm{NP}^A)$ may be relativized either way. However, we do not know if there exists an oracle which separates $\leq_{tt}^p(\mathrm{NP}^A)$ and $\leq_T^p(\mathrm{NP}^A)$ and collapses the Boolean hierarchy over NP. Recall that $\mathrm{NP}(i)$ is the i-th level of the Boolean hierarchy over NP.

Theorem 10 *There is a recursive oracle A such that for all $i \geq 1$, $\mathrm{NP}^A(i) \neq \mathrm{NP}^A(i+1)$ while $\mathrm{PSPACE}^A = \leq_{tt}^p(\mathrm{NP}^A)$.*

Proof Let S^A be a set many-one complete for NP^A under logspace reductions. To obtain a set A with $\mathrm{PSPACE}^A = \leq_{tt}^p(S^A)$, we can modify the construction in [8] of an oracle A which separates the finite levels of the Boolean hierarchy as follows: spread out the "blocks" used in that construction and intersperse a coding of a set U which is PSPACE^A-complete, ensuring that for each y of length i,

$$\begin{aligned} y \in U \leftrightarrow (\exists z)(i^3 - i + 1 \leq |z| \leq i^3 \text{ and } yz \in A) \text{ and} \\ \text{the minimum length of such a } z \text{ is odd.)} \end{aligned}$$

We omit the details which are similar to the proof in [8] of an oracle which separates the first k levels of the Boolean hierarchy and collapses PSPACE to the k-th level. $\square$

An oracle A is a subset of $\{0,1\}^*$; if $\sigma \in A$, then $|\sigma|$ is the length of σ and σ_i denotes the i-th symbol in the string σ. Following Goldsmith and Joseph [9], define a *relativized Boolean formula* to be of the form

$$\Phi^A \ = \ \phi(x_1, \ldots, x_n) \ \wedge \ \left(\bigwedge_{1 \leq i \leq m} \rho_i \in A \right) \ \wedge \ \left(\bigwedge_{1 \leq i \leq k} \tau_i \notin A \right)$$

where ϕ is in conjunctive normal form and each ρ_i and τ_i is a concatenation of the form $z_1 z_2 \cdots z_r$ with each z_j a literal x_j or $\overline{x}_j$. A *truth assignment* α is a mapping from $\{x_i : 0 < i\}$ to $\{0,1\}$ where $\alpha(x_i) = 1$ iff x_i is assigned truth value "True". Given an oracle $A \subset \{0,1\}^*$ we define the truth value of Φ^A under a truth assignment in the obvious way by interpreting $z_1 \cdots z_r \in A$ as meaning $\alpha(z_1) \cdots \alpha(z_r) \in A$. Now define SAT^A to be the set of satisfiable relativized Boolean formulas. It was shown by Goldsmith and Joseph that SAT^A is many-one complete for NP^A for all A. We say that a truth assigment α satisfying Φ^A above is *odd* iff $\alpha(x_n) = 1$. Let $\mathrm{ODDMAXSAT}^A$ be the set of satisfiable relativized Boolean formulas whose maximum satisfying assignment is odd; $\mathrm{ODDMAXSAT}^A$ is evidently in $\leq_T^p(\mathrm{NP}^A)$.

Using the notion of relativized Boolean formula, it is possible to extend Theorems 1-9 to relativized computation. By the relativized version of Theorem 8, we have that a set B is $\leq_{tt}^p(\mathrm{SAT}^A)$ if and only if there is a normalized truth-table reduction of B to SAT^A; ie, if and only if there is a polynomial time computable function f such that for each x, $f(x) = \left\langle y_1, \ldots, y_{n(x)} \right\rangle$ where each y_j encodes a relativized Boolean formula and $\chi_{\mathrm{SAT}^A}(y_1) \geq \chi_{\mathrm{SAT}^A}(y_2) \geq \cdots \geq \chi_{\mathrm{SAT}^A}(y_{n(x)})$ and

$$x \in B \ \leftrightarrow \ \left|\{j : y_j \in \mathrm{SAT}^A\}\right| \text{ is odd.}$$

Theorem 11 *There is a recursive oracle A such that for all $i \geq 1$, $\mathrm{NP}^A(i) \neq \mathrm{NP}^A(i+1)$ and $\leq_{tt}^p(\mathrm{NP}^A) \neq \leq_T^p(\mathrm{NP}^A)$.*

Proof We will construct an oracle A such that $\mathrm{ODDMAXSAT}^A$ is in $\leq_T^p(\mathrm{NP}^A)$ but not in $\leq_{tt}^p(\mathrm{NP}^A)$. The steps in our construction can be interspersed with those of the Cai-Hemachandra construction to ensure that $\mathrm{NP}^A(i) \neq \mathrm{NP}^A(i+1)$ for all i. The latter details are omitted.

Let $\{T_i\}_{i \in \mathbb{N}}$ be an enumeration of all polynomial time transducers computing a function f_i such that for each x, $f_i(x) = \left\langle y_1, \ldots, y_{n(x)} \right\rangle$ where each y_j encodes a relativized Boolean formula. We shall construct the oracle A so that none of the functions f_i is a normalized truth-table reduction of $\mathrm{ODDMAXSAT}^A$ to SAT^A.

Let each T_i compute f_i with polynomial time bound $p_i(n)$; without loss of generality, $p_i(n) \geq n$. Let $\Sigma = \{0,1\}$; so Σ^n is the set of all strings of 0's and 1's of length n. For each n, let Φ_n be the relativized Boolean formula

$$\Phi_n \ = \ (x_1 \vee \neg x_1) \wedge x_1 x_2 \cdots x_n \in A$$

clearly there is a polynomial $t(n)$ such that Φ_n has length $\leq t(n)$. The set A will be constructed in stages; A_s will be the set constructed at stage s and A will be the union of the A_s's. At each stage s, we will select a length $n = n_s$ and form A_s by adding strings of length n to A_{s-1}; no strings are ever removed from A. During stage s there will be a set S of strings available for addition to A. Our construction will maintain the condition that there will always be both even and odd strings available which are lexicographically larger than any string already in A. The goal at stage s is to use Φ_n to witness the fact that f_{s-1} will not be a normalized truth-table reduction of $\mathrm{ODDMAXSAT}^A$ to SAT^A. For any oracle B define h_B so that if $f_{s-1}(\Phi_n) = \langle y_1, \ldots, y_k \rangle$, then $h_B(\Phi_n) = \left|\{j : y_j \in \mathrm{SAT}^B\}\right| \bmod 2$; thus a normalized truth-table reduction f_s to SAT^B causes Φ^n to be accepted iff $h_B(\Phi_n) = 1$. Our goal at stage s is to arrange that $\Phi_n \in \mathrm{ODDMAXSAT}^A$ iff $h_A(\Phi_n) = 0$.

To begin the construction of A, set $A_0 = \emptyset$ and $n_0 = 0$. Now construct A_{s+1} as follows: Let $q(n) = p_s(t(n))$. Choose $n = n_{s+1} > 2^{n_s}$ big enough so that $4q(n) < 2^{n-1}$. Run T_s on input Φ_n; the output $f_s(\Phi_n)$ is a sequence $\langle y_1, \ldots, y_k \rangle$ of total length $\leq q(n)$ with each y_j encoding a relativized Boolean formula of the form $\phi_j \wedge \bigwedge \rho_i \in A \wedge \bigwedge \tau_i \notin A$. We may assume that $\chi_{\mathrm{SAT}^{A_s}}(y_j) \geq \chi_{\mathrm{SAT}^{A_s}}(y_{j+1})$ for all j; otherwise, we let $A_{s+1} = A_s$ and f_s will not be a normalized truth-table reduction to SAT^A. Likewise, we may assume that for all A_{s+1} formed by adding some set of strings of length n to A_s we have $\chi_{\mathrm{SAT}^{A_{s+1}}}(y_j) \geq \chi_{\mathrm{SAT}^{A_{s+1}}}(y_{j+1})$ for all j; otherwise we pick an A_{s+1} such that f_s will not be a normalized truth-table reduction to $\mathrm{SAT}^{A_{s+1}}$. Let b_j be the number of conjuncts of the form $\tau_i \notin A$ in the formula coded by y_j. Let X be the set $\{j : y_j \in \mathrm{SAT}^{A_s}\}$ and then $h_{A_s}(\Phi_n) = (|X| \bmod 2)$. If $h_{A_s}(\Phi_n) = 1$ then we may set $A_{s+1} = A_s$; in this case, $h_A(\Phi_n) = h_{A_{s+1}}(\Phi_n) = 1$ but A will contain no strings of length n and hence $\Phi_n \notin \mathrm{ODDMAXSAT}^A$.

So suppose that X has even cardinality. We initialize S to be the set of all strings of length n — these are the strings available to be added to A at this stage. Also initialize A_{s+1} to be A_s. S will be the set of strings available to be added to A_{s+1}; at any point in the construction, each string in S will be lexicographically greater than the strings already in A_{s+1}. To build up A_{s+1} we run the following procedure repeatedly as necessary:

Loop:

For each $j \in X$ (if any) let α_j be a fixed satisfying assignment of y_j with respect to the oracle A_{s+1} constructed so far. The formula coded by y_j may contain clauses "$\tau \notin A$" which under the satisfying assignment α_j specify a string as being out of A. Let N_j be the set of strings that α_j and y_j specify as being out of A; note $b_j \geq |N_j|$. (If α_j and N_j have already been assigned in an earlier iteration of the loop, leave them unchanged.)

Set $L := \bigcup_{j \in X} N_j$. So $|L| \leq \sum_{j \in X} b_j \leq q(n)$.

Set $S := S - L$. Note that adding members of S to A_{s+1} will never unsatisfy any member y_j of $f_s(\Phi_n)$.

Let $S' := \{\sigma \in S : \sigma_n = 1 \leftrightarrow h_{A_{s+1}}(\Phi_n) = 0\}$; or, in words, S' is the set of odd (even) strings in S if $h_{A_{s+1}}(\Phi_n)$ is 0 (resp., 1). Since strings in S are greater than strings in A_{s+1}, if we can add a member of S' to A_{s+1} without altering the satisfiability of the members of $f_s(\Phi_n)$ then this will force f_s to not be a truth-table reduction of $\mathrm{ODDMAXSAT}^{A_{s+1}}$ to $\mathrm{SAT}^{A_{s+1}}$.

If there is some member σ of S' such that adding σ to A_{s+1} does not change the satisfiability of the members y_j of $f_s(\Phi_n)$, then set $A_{s+1} := A_{s+1} \cup \{\sigma\}$ and exit from the loop. In this case, we have that $\Phi_n \in \text{ODDMAXSAT}^{A_{s+1}}$ iff $h_{A_{s+1}}(\Phi_n) = 0$. Note that this case always applies when $|X| = k$ and every member of $f_s(\Phi_n)$ is already satisfied, since the exclusion of the set L guarantees that nothing becomes unsatisfied.

Otherwise, add the lexicographically minimum element σ of S' to A_{s+1}. This will satisfy at least one more y_j; so redefine X to be $\{j : y_j \in \text{SAT}^{A_{s+1}}\}$ and redefine $h_{A_{s+1}}(\Phi_n) := |X| \bmod 2$. Note that the cardinality of X is always increased. Finally, remove from S the string σ and every string σ' which is lexicographically less than σ. (It is important that σ be the least element of S' so that not too many small σ''s need to be removed from S.) Now repeat the loop.

Endloop

To see that the procedure halts there are two things to notice. Firstly, the loop will be iterated at most k times since $|X|$ increases each time and the loop halts after $|X| = k$. Secondly, the set S always contains both even and odd members by our choice of n, since it is easy to see that at most $2 \cdot |L| + 2 \cdot k \leq 4q(n)$ strings can be removed from S, where $|L|$ means the cardinality of the final set L at the end of the procedure. □

A stronger notion of relativized polynomial truth-table reductions may be defined by letting the reduction query the oracle A; we denote this reducibility by $\leq_{tt}^{\mathrm{P}^A}$. It is easy to modify the above construction to ensure that it is not the case that $\text{ODDMAXSAT}^A \leq_{tt}^{\mathrm{P}^A} SAT^A$ by freezing any queries that f_s makes of length n_{s+1} out of A_{s+1}; it will be necessary to choose n_{s+1} large enough so that S is big enough after the 'frozen' strings are removed. The result is an oracle A such that $\leq_{tt}^{\mathrm{P}^A}(\mathrm{NP}^A) \neq \leq_T^p(\mathrm{NP}^A)$.

5 Conclusions

To summarize, we list most of the characterizations of $\leq_{tt}^p(\mathrm{NP})$ discussed in this paper:

Theorem 12 *The following are equivalent:*

- *A is polynomial time (Boolean circuit) truth-table reducible to* SAT*,*
- *A is polynomial time, Boolean formula truth-table reducible to* SAT*,*
- *A is logspace, Boolean formula truth-table reducible to* SAT*,*
- *A is logspace, Turing reducible to* SAT*,*
- *A is definable by a $\Sigma_2^b \cap \Pi_2^b$ formula,*
- *A is in Σ_2^L (equivalently, Π_2^L),*
- *A is in* $\mathrm{P}^{\mathrm{SAT}}[O(\log n)]$*.*
- *A is recognizable in polynomial time with a constant number of rounds of parallel queries to* SAT*.*

The multiplicity of equivalent definitions for $\leq_{tt}^p(\mathrm{NP})$ indicates that it is a robust concept and hence likely to be a natural and useful class for theoretical computational complexity.

References

[1] R. Beigel, *Bounded queries to SAT and the Boolean hierarchy.* to appear in Theoretical Computer Science.

[2] S. R. Buss, *Bounded Arithmetic*, Bibliopolis, 1986. Revision of 1985 Princeton University Ph.D. thesis.

[3] ——, *Axiomatizations and conservation results for fragments of bounded arithmetic*, in Logic and Computation, proceedings of a Workshop held Carnegie-Mellon University, 1987, vol. 106 of Contemporary Mathematics, American Mathematical Society, 1990, pp. 57–84.

[4] S. R. Buss and L. Hay, *On truth-table reducibility to SAT and the difference hierarchy over NP*, in Proceedings of the Structure in Complexity Conference, June 1988, pp. 224–233.

[5] J. Cai, T. Gundermann, J. Hartmanis, L. Hemachandra, V. Sewelson, K. Wagner, and G. Wechsung, *The Boolean hierarchy I: Structural properties*, SIAM Journal on Computing, 17 (1988), pp. 1232–1252.

[6] ——, *The Boolean hierarchy II: Applications*, SIAM Journal on Computing, 18 (1989), pp. 95–111.

[7] J.-y. Cai and L. Hemachandra, *The Boolean hierarchy: Hardware over NP*, Tech. Rep. #TR85-724, Cornell University, Computer Science Department, December 1985.

[8] ——, *The Boolean hierarchy: Hardware over NP*, in Structure in Complexity, Lecture Notes in Computer Science #223, Springer Verlag, 1986, pp. 105–124.

[9] J. Goldsmith and D. Joseph, *Three results on polynomial isomorphism of complete sets*, in Proceedings of the 27th Annual Symposium on Foundations of Computer Science, IEEE Computer Society, Oct. 1986, pp. 390–397.

[10] F. Hausdorff, *Set Theory*, Chelsea, third ed., 1978.

[11] L. A. Hemachandra, *The strong exponential hierarchy collapses*, in Proceedings 19-th Annual ACM Symposium on Theory of Computing, 1987, pp. 110–122. To appear in JCSS.

[12] B. Jenner, B. Kirsig, and K.-J. Lange, *The logarithmic alternation hierarchy collapses: $A\Sigma_2^{\mathcal{L}} = A\Pi_2^{\mathcal{L}}$*, Information and Computation, 80 (1989), pp. 269–288.

[13] J. Köbler, U. Schöning, and K. W. Wagner, *The difference and truth-table hierarchies for NP*, Informatique Theorique et Applications, 21 (1987), pp. 419–435.

[14] M. W. Krentel, *The complexity of optimization problems*, Journal of Computer and System Sciences, 36 (1988), pp. 490–509.

[15] R. E. Ladner, *The circuit value problem is log space complete for P*, SIGACT News, 7 (1975), pp. 18–20.

[16] R. E. Ladner and N. A. Lynch, *Relativization of questions about log space computability*, Math. Systems Theory, 10 (1976), pp. 19–32.

[17] R. E. Ladner, N. A. Lynch, and A. L. Selman, *A comparison of polynomial time reducibilities*, Theoretical Comput. Sci., 1 (1975), pp. 103–123.

[18] N. A. Lynch, *Log space recognition and translation of parenthesis languages*, J. Assoc. Comput. Mach., 24 (1977), pp. 583–590.

[19] K. W. Wagner, *More complicated questions about maxima and minima, and some closures of NP*, vol. 51, 1987, pp. 53–80.

[20] ——, *On restricting the access to an NP oracle*, in Automata, Languages and Programming, Lecture Notes in Computer Science #317, Springer-Verlag, 1988, pp. 682–696.

[21] ——, *Bounded query classes*, SIAM Journal on Computing, 19 (1990), pp. 833–846.

[22] G. Wechsung, *On the Boolean closure of NP*, in Proc. Int'l Conf. on Fundamentals Computation Theory, Lecture Notes in Computer Science #199, 1985, pp. 485–493. Co-authored by K. Wagner.