

## On Truth-Table Reducibility to SAT

SAMUEL R. BUSS\*

*Department of Mathematics,  
University of California, San Diego,  
La Jolla, California 92093*

AND

LOUISE HAY†

*Department of Mathematics, Statistics, and Computer Science,  
University of Illinois at Chicago,  
Chicago, Illinois*

DEDICATED TO THE MEMORY OF LOUISE HAY

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  $\mathcal{A}_2^?$  is not equal to the class of predicates polynomial time truth-table reducible to SAT. © 1991 Academic Press, Inc.

### 1. INTRODUCTION

Ladner, Lynch, and Selman (1975) 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 (1976) this is equivalent to the notion of polynomial 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

\* 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.

equivalent to logspace Turing reducibility to SAT. Several other equivalent definitions are given for the 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 (1986), but answer only a special case of the original questions of Ladner and Lynch.

Let  $\leq_{tt}^p(\text{NP})$  denote the class of decision problems polynomial time truth-table reducible to NP; this paper presents several other characterizations of this class. First, Krentel (1986) defines the class  $\text{P}^{\text{SAT}}[O(\log n)]$ ; we show this is equal to  $\leq_{tt}^p(\text{NP})$ . Cai and Hemachandra (1985) defined a hierarchy  $\Sigma_k^L, \Pi_k^L$  above the finite levels of the difference hierarchy over NP; we show this hierarchy collapses with  $\Sigma_2^L = \Pi_2^L = \leq_{tt}^p(\text{NP})$ . Furthermore,  $\leq_{tt}^p(\text{NP})$  is precisely the set of predicates definable by  $\Sigma_2^b \cap \Pi_2^b$  formulas as defined by Buss [2].

The class  $\leq_{tt}^p(\text{NP})$  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 permissible—so a polynomial time Turing machine which makes  $O(1)$  rounds of parallel queries to SAT recognizes a  $\leq_{tt}^p(\text{NP})$  predicate.

In the final section we provide evidence that  $\leq_{tt}^p(\text{NP})$  is distinct from other commonly studied classes in computational complexity by giving one oracle relative to which  $\leq_{tt}^p(\text{NP})$  is equal to PSPACE and another oracle relative to which  $\leq_{tt}^p(\text{NP})$  and  $\Delta_2^p$  are distinct. For both oracles, the Boolean hierarchy over NP is proper and hence distinct from  $\leq_{tt}^p(\text{NP})$  and, in particular,  $\leq_{tt}^p(\text{NP})$  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 (1988a, 1988b). 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 (1987). Similar “counting” techniques have been used recently by a number of authors; an early example is Jenner, Kirsig, and Lange (1989); the work of Beigel (1987) was particularly helpful to us in the original formulation of our results.

The current paper is a slightly revised version of part of Buss and Hay (1987); 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 ( $\leq_u^p$ ) was first introduced by Ladner, Lynch, and Selman (1975) and further studied by Ladner and Lynch (1976). The general idea is that  $A \leq_u^p B$  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, 1986, and Köbler, Schöning, and Wagner, 1986.)

2.1. *Some Definitions and a Main Theorem*

As usual, P and NP denote the classes of polynomial time and non-deterministic 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  $\text{eval}(z, x_1, \dots, x_n)$  be the value of the Boolean function coded by  $z$  on binary inputs  $x_1, \dots, x_n$ . Let  $\chi_B$  be the characteristic function of  $B$ , so  $\chi_B(y) = 1$  if  $y \in B$  and  $\chi_B(y) = 0$  otherwise. Then:

(1)  $A \leq_u^p B$ ,  $A$  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  $x$ ,  $f(x) = \langle z, y_1, \dots, y_k \rangle$  with  $z$  coding a Boolean circuit such that

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

(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  $x$ ,  $f(x) = \langle z, y_1, \dots, y_k \rangle$  with  $z$  coding a Boolean formula such that

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

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

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

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

Ladner, Lynch, and Selman (1975) 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  $\leq_{ft}^p(A)$  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  $\leq_{ft}^p(\text{NP})$  denote  $\leq_{tt}^p(\text{SAT})$ ,  $\leq_{bf}^p(\text{SAT})$ , and  $\leq_{ft}^p(\text{SAT})$ , 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  $\leq_{ft}^{\log}$  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 (1976) 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 (1977) that Boolean formulas can be evaluated in logspace, if  $A \leq_{bf}^{\log} B$  then  $A \leq_{LL}^{\log} B$ ; and by Ladner’s (1975) theorem 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  $\leq_{tt}^p(\text{NP}) \supseteq \leq_{bf}^p(\text{NP}) \supseteq \leq_{ft}^p(\text{NP})$ ; complete problems for these classes are given by Köbler, Schöning, and Wagner (1986) and in Wagner (1986). The next theorem states that  $\leq_{tt}^p(\text{NP})$  is actually equal to  $\leq_{bf}^p(\text{NP})$  and the following theorem and corollary show that they are equal to all of  $\leq_{bf}^{\log}(\text{NP})$ ,  $\leq_{LL}^{\log}(\text{NP})$ ,  $\leq_{tt}^{\log}(\text{NP})$ , and  $\leq_T^{\log}(\text{NP})$ . This theorem was independently discovered by K. Wagner.

THEOREM 1.  $\leq_{tt}^p(\text{NP}) = \leq_{bf}^p(\text{NP})$ .

*Proof.* Let  $\Sigma$  be a finite alphabet and suppose  $C \subseteq \Sigma^*$  and  $C \leq_{tr}^p \text{SAT}$ ; let  $f$  be a polynomial time function giving a truth-table reduction of  $C$  to  $\text{SAT}$ . For all  $x \in \Sigma^*$ ,  $f(x)$  codes a sequence  $\langle c, f_1(x), \dots, f_k(x) \rangle$ ; we shall sometimes write  $c(x)$  and  $k(x)$  to make it 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  $\text{SAT}(y)$  is the predicate “ $y$  is satisfiable,” also let  $\text{TRU}(u, y)$  be the polynomial time predicate “ $u$  codes a satisfying assignment of  $y$ ”; so  $\text{SAT}(y) \leftrightarrow (\exists u < y) \text{TRU}(u, y)$ . Let  $B(i, x)$  be the predicate

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

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

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

So  $A(i, x)$  is in  $\text{NP}$  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  $f_j(x) \in \text{SAT}$ . Hence  $C$  can be defined by

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

Finally, we claim that this puts  $C$  in  $\leq_{tr}^p(\text{SAT})$ . 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  $\text{SAT}$  is  $\text{NP}$ -complete, there are polynomial time functions  $f_A$  and  $f_B$  so that  $f_A(i, x) \in \text{SAT}$  iff  $A(i, x)$  and similarly for  $B$ . Hence for any  $x$ ,  $x \in C$  if and only if the Boolean formula

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

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  $\text{SAT}$ . Q.E.D.

Actually the above proof gives a stronger result:

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

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

**COROLLARY 3.** *A decision problem  $C$  is in  $\leq_{tt}^p(\text{NP})$  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 (1976) 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. ■

**THEOREM 4.** *Every predicate in  $\leq_{tt}^p(\text{NP})$  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*

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

*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  $\leq_{tt}^p(\text{NP})$  is closed under complementation. ■

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

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

First we provide a syntactic characterization of formulas which define predicates in  $\leq_{tt}^p(\text{NP})$  by showing that a decision problem is in  $\leq_{tt}^p(\text{NP})$  if and only if it is definable by a  $\Sigma_2^b \cap \Pi_2^b$ -formula in the sense of Buss (1986). The superscript  $b$  is used to indicate that  $\Sigma_2^b$  and  $\Pi_2^b$  are classes of formulas; the  $\Sigma_2^b$  (resp. the  $\Pi_2^b$ ) formulas define precisely the  $\Sigma_2^p$  (resp., the  $\Pi_2^p$ ) predicates. Hence we shall establish that a decision problem is in  $\leq_{tt}^p(\text{NP})$  if and only if it has a defining formula which is simultaneously and explicitly in both  $\Sigma_2^b$  and  $\Pi_2^b$ . (However, this is in no way intended to say that  $\Sigma_2^p \cap \Pi_2^p = \leq_{tt}^p(\text{NP})$ .)

Buss (1986) 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  $\Sigma_i^b$  and  $\Pi_i^b$  are classes of bounded formulas defined by counting alternations of (bounded) quantifiers, ignoring the sharply bounded ones. The formulas in the class  $\Sigma_i^b$  or  $\Pi_i^b$  define precisely the predicates in the  $i$ th level of the polynomial time hierarchy  $\Sigma_i^p$  or  $\Pi_i^b$ , respectively. The class  $\Sigma_2^b \cap \Pi_2^b$  has already been studied by Buss (1987) who showed that it is the largest class of formulas for which  $S_2^1$  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  $(\forall y \leq p(|\mathbf{x}|))$  or  $(\exists y \leq p(|\mathbf{x}|))$  for  $p$  a polynomial and  $|\mathbf{x}|$  the total length of the binary representations of  $\mathbf{x}$ . An *NP-formula* is a formula which defines a predicate in NP.  $\Sigma_2^b \cap \Pi_2^b$  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  $\leq_u^p(\text{NP})$  is equal to the class of predicates definable by  $\Sigma_2^b \cap \Pi_2^b$  formulas.*

*Proof.* Theorem 4 states that every predicate in  $\leq_u^p(\text{NP})$  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  $\Sigma_2^b \cap \Pi_2^b$  formula. For the converse, it is easy to prove by induction on the complexity of formulas that every  $\Sigma_2^b \cap \Pi_2^b$  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  $\Psi$  is a Boolean combination of NP formulas. Since SAT is NP-complete we may assume that  $\Psi$  is a Boolean combination of formulas of the form  $\text{SAT}(f(x, y))$  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

$$\bigwedge_{y_1=0}^{p(|x|)} \bigwedge_{y_2=0}^{p(|x|)} \cdots \bigwedge_{y_k=0}^{p(|x|)} \Psi,$$

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

Cai and Hemachandra (1985) define a hierarchy of sets denoted  $\Sigma_k^L, \Pi_k^L$  for  $k \geq 0$ . They defined these sets in terms of Turing machine computation: a predicate is in  $\Sigma_k^L$  (or  $\Pi_k^L$ ) 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  $\Sigma$  be an alphabet. A predicate  $A \subseteq \Sigma^*$  is in  $\Sigma_k^L$  if and only if  $A$  can be defined as the set of  $x \in \Sigma^*$  such that

$$(\exists y_1 \leq p_1(|x|))(\forall y_2 \leq p_2(|x|)) \cdots (Q_k y_k \leq p_k(|x|)) \\ [(R(x, \mathbf{y}) \rightarrow B(x, \mathbf{y})) \wedge (\neg R(x, \mathbf{y}) \rightarrow C(x, \mathbf{y}))],$$

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  $\Pi_k^L$  predicate if and only if the complement of  $A$  is a  $\Sigma_k^L$  predicate.

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

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

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

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

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

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

Hence  $C$  is in  $\Sigma_2^L$ . Since  $\leq_{IT}^p(\text{NP})$  is closed under complementation,  $\leq_{IT}^p(\text{NP})$  is also a subset of  $\Pi_2^L$  and we are done. ■



### 2.3. Another Characterization of $\leq_p^u(\text{NP})$

Krentel (1986) defines  $\text{P}^{\text{SAT}}[O(\log n)]$  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.**  $\text{P}^{\text{SAT}}[O(\log n)] = \leq_p^u(\text{NP})$ .

This theorem is well known but we include its proof for the sake of completeness. Hemachandra (1987) has already established the harder ( $\supseteq$ ) direction; we first found the proof using ideas from Beigel (1987). The method of proof is also used in Section 4.

*Proof.* ( $\subseteq$ ) Suppose  $A \in \text{P}^{\text{SAT}}[O(\log n)]$  and  $A$  is recognized by Turing machine  $M^{\text{SAT}}$  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  $\Omega$ . 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  $M^{\text{SAT}}(x)$  in terms of the answers to the queries to SAT.

( $\supseteq$ ) Let  $C$  be in  $\leq_p^u(\text{NP})$  and let  $A(i, x)$  and  $B(i, x)$  be as in the proof of Theorem 1, so that, for all  $x$ ,

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

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  $B(i, x)$ ,

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

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,

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

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. ■

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 Buss and Hay (1989).

**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  $x$ ,  $f(x) = \langle y_1, \dots, y_{k(x)} \rangle$  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  $\leq_{tt}^p(\text{NP})$  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 (1986) under the name “truth-table Hausdorff reductions” and Theorem 8 is also a direct consequence of Theorem 2 and a theorem of Wagner’s that  $\leq_{bf}^p(\text{NP})$  equals the set of predicates which are polynomial time, truth-table Hausdorff reducible to an NP set.

### 3. MULTIPLE ROUNDS OF PARALLEL QUERIES

Ladner, Lynch, and Selman (1975) proved that  $\leq_{tt}^p(\text{NP})$  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  $\leq_{IT}^p(\text{NP})$ , 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  $\geq 1$ . 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  $\text{TRU}(x, y)$  is the predicate “ $x$  is a satisfying assignment for  $y$ .” A  $k$ -tuple of integers is a sequence  $\mathbf{r} = \langle r_1, \dots, r_k \rangle$  of integers. We order the  $k$ -tuples lexicographically so  $\mathbf{r} < \mathbf{s}$  iff for some  $i$ ,  $r_i < s_i$  and  $(\forall j < i)(r_j = s_j)$ .

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

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

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

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

Finally, note that the  $k$  in Theorem 9 must be constant. If  $k$  is unrestricted we get all of  $\mathcal{A}_2^p$ , 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  $\leq_{it}^p(\text{NP})$ . 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  $\leq_{it}^p(\text{NP})$  includes NP and is a subset of  $\Delta_2^p = \leq_T^p(\text{NP})$ ; it is of course open whether these three classes are distinct since, if so, then  $\text{P} \neq \text{NP}$ . 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 (1978) and the Boolean hierarchy over NP has been extensively studied by a number of researchers (Wechsung, 1985; Cai and Hemachandra, 1986; Köbler *et al.*, 1986; Cai *et al.*, 1988, 1989). Cai and Hemachandra (1986) showed that there are oracles relative to which the Boolean hierarchy over NP has infinitely many levels, and oracles relative to which the Boolean hierarchy extends exactly  $k$  levels, with PSPACE collapsing to the  $(k + 1)$ th level. We extend these results by showing that in relativized worlds where the Boolean hierarchy over NP has infinitely many levels, the condition of  $\leq_{it}^p(\text{NP}^A)$  being equal to  $\leq_T^p(\text{NP}^A)$  may be relativized either way. However, we do not know if there exists an oracle which separates  $\leq_{it}^p(\text{NP}^A)$  and  $\leq_T^p(\text{NP}^A)$  and collapses the Boolean hierarchy over NP. Recall that  $\text{NP}(i)$  is the  $i$ th level of the Boolean hierarchy over NP.

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

*Proof.* Let  $S^A$  be a set many-one complete for  $\text{NP}^A$  under logspace reductions. To obtain a set  $A$  with  $\text{PSPACE}^A = \leq_{it}^p(S^A)$ , we can modify the construction in (Cai and Hemachandra, 1986) 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  $\text{PSPACE}^A$ -complete, ensuring that for each  $y$  of length  $i$ ,

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

We omit the details which are similar to the proof in *op. cit.* of an oracle which separates the first  $k$  levels of the Boolean hierarchy and collapses PSPACE to the  $k$ th level. ■

An oracle  $A$  is a subset of  $\{0, 1\}^*$ ; if  $\sigma \in A$ , then  $|\sigma|$  is the length of  $\sigma$  and  $\sigma_i$  denotes the  $i$ th symbol in the string  $\sigma$ . Following Goldsmith and Joseph (1986), define a *relativized Boolean formula* to be of the form

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

where  $\phi$  is in conjunctive normal form and each  $\rho_i$  and  $\tau_i$  is a concatenation of the form  $z_1 z_2 \dots z_r$  with each  $z_j$  a literal  $x_j$  or  $\bar{x}_j$ . A *truth assignment*  $\alpha$  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  $\Phi^A$  under a truth assignment in the obvious way by interpreting  $z_1 \dots z_r \in A$  as meaning  $\alpha(z_1) \dots \alpha(z_r) \in A$ . Now define  $\text{SAT}^A$  to be the set of satisfiable relativized Boolean formulas. It was shown by Goldsmith and Joseph that  $\text{SAT}^A$  is many-one complete for  $\text{NP}^A$  for all  $A$ . We say that a truth assignment  $\alpha$  satisfying  $\Phi^A$  above is *odd* iff  $\alpha(x_n) = 1$ . Let  $\text{ODDMAXSAT}^A$  be the set of satisfiable relativized Boolean formulas whose maximum satisfying assignment is odd;  $\text{ODDMAXSAT}^A$  is evidently in  $\leq_T^p(\text{NP}^A)$ .

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  $\leq_{\text{m}}^p(\text{SAT}^A)$  if and only if there is a normalized truth-table reduction of  $B$  to  $\text{SAT}^A$ ; i.e., if and only if there is a polynomial time computable function  $f$  such that for each  $x$ ,  $f(x) = \langle y_1, \dots, y_{n(x)} \rangle$ , where each  $y_j$  encodes a relativized Boolean formula and  $\chi_{\text{SAT}^A}(y_1) \geq \chi_{\text{SAT}^A}(y_2) \geq \dots \geq \chi_{\text{SAT}^A}(y_{n(x)})$  and

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

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

*Proof.* We will construct an oracle  $A$  such that  $\text{ODDMAXSAT}^A$  is in  $\leq_T^p(\text{NP}^A)$  but not in  $\leq_{\text{m}}^p(\text{NP}^A)$ . The steps in our construction can be interspersed with those of the Cai–Hemachandra construction to ensure that  $\text{NP}^A(i) \neq \text{NP}^A(i+1)$  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  $x$ ,  $f_i(x) = \langle y_1, \dots, y_{n(x)} \rangle$ , 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  $\text{ODDMAXSAT}^A$  to  $\text{SAT}^A$ .

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  $\Sigma^n$  is the set of all strings of 0's and 1's of length  $n$ . For each  $n$ , let  $\Phi_n$  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  $\Phi_n$  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  $\Phi_n$  to witness the fact that  $f_{s-1}$  will not be a normalized truth-table reduction of  $\text{ODDMAXSAT}^A$  to  $\text{SAT}^A$ . For any oracle  $B$  define  $h_B$  so that if  $f_{s-1}(\Phi_n) = \langle y_1, \dots, y_k \rangle$ , then  $h_B(\Phi_n) = |\{j : y_j \in \text{SAT}^B\}| \bmod 2$ ; thus a normalized truth-table reduction  $f_s$  to  $\text{SAT}^B$  causes  $\Phi_n$  to be accepted iff  $h_B(\Phi_n) = 1$ . Our goal at stage  $s$  is to arrange that  $\Phi_n \in \text{ODDMAXSAT}^A$  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  $\Phi_n$ ; the output  $f_s(\Phi_n)$  is a sequence  $\langle y_1, \dots, y_k \rangle$  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  $\chi_{\text{SAT}^A}(y_j) \geq \chi_{\text{SAT}^A}(y_{j+1})$  for all  $j$ ; otherwise, we let  $A_{s+1} = A_s$  and  $f_s$  will not be a normalized truth-table reduction to  $\text{SAT}^A$ . 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  $\chi_{\text{SAT}^{A_{s+1}}}(y_j) \geq \chi_{\text{SAT}^{A_{s+1}}}(y_{j+1})$  for all  $j$ ; otherwise we pick an  $A_{s+1}$  such that  $f_s$  will not be a normalized truth-table reduction to  $\text{SAT}^{A_{s+1}}$ . 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  $\{j : y_j \in \text{SAT}^{A_s}\}$  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  $\Phi_n \notin \text{ODDMAXSAT}^A$ .

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  $\alpha_j$  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  $\alpha_j$  specify a string as being out of  $A$ . Let  $N_j$  be the set of strings that  $\alpha_j$  and  $y_j$  specify as being out of  $A$ ; note  $b_j \geq |N_j|$ . (If  $\alpha_j$  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  $\text{ODDMAXSAT}^{A_{s+1}}$  to  $\text{SAT}^{A_{s+1}}$ .

If there is some member  $\sigma$  of  $S'$  such that adding  $\sigma$  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 exist 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  $\sigma$  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  $\sigma$  and every string  $\sigma'$  which is lexicographically less than  $\sigma$ . (It is important that  $\sigma$  be the least element of  $S'$  so that not too many small  $\sigma$ 's need to be removed from  $S$ .) Now repeat the loop.

*Endloop*

To see that the procedure halts there are two things to notice. First, the loop will be iterated at most  $k$  times since  $|X|$  increases each time and the loop halts after  $|X| = k$ . Second, 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  $\leq_{tt}^{P^A}$ . It is easy to modify the above construction to ensure

that it is not the case that  $\text{ODDMAXSAT}^A \leq_{tt}^{p^A} \text{SAT}^A$  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  $\leq_{tt}^{p^A}(\text{NP}^A) \neq \leq_T^p(\text{NP}^A)$ .

## 5. CONCLUSIONS

To summarize, we list most of the characterizations of  $\leq_{tt}^p(\text{NP})$  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  $\Sigma_2^L$  (equivalently,  $\Pi_2^L$ ),*
- *$A$  is in  $\text{P}^{\text{SAT}}[O(\log n)]$ .*
- *$A$  is recognizable in polynomial time with a constant number of rounds of parallel queries to SAT.*

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

RECEIVED April 27, 1988; FINAL MANUSCRIPT RECEIVED October 27, 1989

## REFERENCES

- BEIGEL, R. (1987), Bounded queries to SAT and the Boolean hierarchy, *Theoretical Computer Science*, to appear.
- BUSS, S. R. (1986), "Bounded Arithmetic," Bibliopolis, Napoli; revision of 1985 Princeton University Ph.D. thesis.
- BUSS, S. R. (1990), Axiomatizations and conservation results for fragments of bounded arithmetic, in "Proceedings, Workshop in Logic and Computation, 1987," *Contemp. Math.* **106**, 57–84.
- BUSS, S. R. AND HAY, L. (1988), On truth-table reducibility to SAT and the difference hierarchy over NP, in "Proceedings, Structure in Complexity Conference, June," pp. 224–233.
- CAI, J., GUNDERMANN, T., HARTMANIS, J., HEMACHANDRA, L., SEWELSON, V., WAGNER, K., AND WECHSUNG, G. (1988), The Boolean hierarchy. I. Structural properties, *SIAM J. Comput.* **17**, 1232–1252.



- CAI, J., *et al.* (1989), The Boolean hierarchy. II. Applications, *SIAM J. Comput.* **18**, 95–111.
- CAI, J.-Y., AND HEMACHANDRA, L. (1985), “The Boolean Hierarchy: Hardware over NP,” Tech. Rep. TR85–724, Cornell University, Computer Science Department, December.
- CAI, J.-Y., AND HEMACHANDRA, L. (1986), The Boolean hierarchy: Hardware over NP, in “Structure in Complexity,” Lecture Notes in Computer Science, Vol. 223, pp. 105–124, Springer-Verlag, New York/Berlin.
- GOLDSMITH, AND JOSEPH, D. (1986), Three results on polynomial isomorphism of complete sets, in “Proceedings, 27th Annual Symposium on Foundations of Computer Science, IEEE, Oct. 1986,” pp. 390–397.
- HAUSDORFF, F. (1978), “Set Theory 3rd ed., Chelsea, New York.
- HEMACHANDRA, L. A. (1989), The strong exponential hierarchy collapses, in “Proceedings, 19th Annual ACM Symposium on Theory of Computing, 1987,” pp. 110–122; *J. Comput. System Sci.* **39**, 299–322.
- JENNER, B., KIRSIG, B., AND LANGE, K.-J. (1989), The logarithmic alternation hierarchy collapses:  $\mathbf{A}\Sigma_2^{\mathcal{L}} = \mathbf{A}\Pi_2^{\mathcal{L}}$ , *Inform. and Comput.* **80** (1989), 269–288.
- KÖBLER, J., SCHÖNING, U., AND WAGNER, K. W. (1987), The difference and truth-table hierarchies for NP, *Informatique Théorique et Applications* **21**, 419–435.
- KRENTEL, M. W. (1986), The complexity of optimization problems, in “Proceedings, 18th Annual ACM Symposium on Theory of Computing,” pp. 69–76.
- LADNER, R. E. (1975), The circuit value problem is log space complete for P, *SIGACT News* **7**, 18–20.
- LADNER, R. E., AND LYNCH, N. A. (1976), Relativization of questions about log space computability, *Math. Systems Theory* **10**, 19–32.
- LADNER, R. E., LYNCH, N. A., AND SELMAN, A. L. A comparison of polynomial time reducibilities, *Theoret. Comput. Sci.* **1**, 103–123.
- LYNCH, N. A. (1977), Log space recognition and translation of parenthesis languages, *J. Assoc. Comput. Mach.* **24**, 583–590.
- WAGNER, K. W. (1987), More complicated questions about maxima and minima, and some closures of NP, *Theoret. Comput. Sci.* **51**, 53–80.
- WAGNER, K. W. (1988a), Bounded query classes, in “Proceedings, Structure in Complexity Theory Conference,” pp. 260–277, IEEE Comput. Washington, DC.
- WAGNER, K. W. (1988b), On restricting the access to an NP oracle, in “Automata, Languages and Programming,” Lecture Notes in Computer Science, Vol. 317, pp. 682–696, Springer-Verlag, New York/Berlin.
- WECHSUNG, G. (1985), On the Boolean closure of NP, in “Proceedings, Int’l Conf. on Fundamentals Computation Theory,” Lecture Notes in Computer Science, Vol. 199, pp. 485–493 (co-authored by K. Wagner), Springer-Verlag, New York/Berlin.