On Truth-Table Reducibility to SAT and the Difference Hierarchy over NP

Samuel R. Buss*
Department of Mathematics
University of California, Berkeley

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

November 1987

Abstract

We show that polynomial time truth-table reducibility via Boolean circuits to SAT is the same as log space truth-table reducibility via Boolean formulas to SAT and the same as log space 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. Finally, we show that the infinite difference hierarchy over NP is equal to Δ_2^p and give an oracle oracle separating Δ_2^p from the class of predicates polynomial time truth-table reducible to SAT.

1 Introduction

*Supported in part by NSF postdoctoral fellowship DMS-8511465.

†Work partially completed while on sabbatical leave at the University of California, Berkeley

Ladner, Lynch, and Selman [21] 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 [20] 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. It is shown below that, for reducibility to the NP-complete problem SAT, polynomial time truth-table reducibility via Boolean circuits is equivalent to log space truth-table reducibility via Boolean formulas. Consequently, it is also equivalent to log space Turing reducibility to SAT. This answers a question raised by Wagner [24] and by Köbler, Schöning and Wagner [18], but only answers 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 [19] defines the class $\mathrm{P}^{\mathrm{SAT}}(O(\log n))$; we show this is equal to $\leq_{tt}^{p}(\mathrm{NP})$. Cai and Hemachandra [5] 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 [4].

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 k rounds of parallel queries to SAT recognizes a $\leq_{tt}^{p}(\mathrm{NP})$ predicate.

Cai and Hemachandra [6], Wagner and Wechsung [26], Wagner [24] and Beigel [2] have studied the finite levels of the difference hierarchy over NP. We study below the infinite difference hierarchy over NP and show that it is precisely Δ_2^p. We provide some new natural complete problems for the finite levels of the difference hierarchy of NP and give another characterization of $\leq_{tt}^{p}(\mathrm{NP})$ in terms of the infinite difference hierarchy over NP.

Finally we give an oracle relative to which Δ_2^p is not equal to $\leq_{tt}^{p}(\mathrm{NP})$. In fact, even for relativizations which separate all the levels of the finite difference hierarchy, the condition $\Delta_2^p = \leq_{tt}^{p}(\mathrm{NP})$ may be relativized either way.

A lot of the results of this paper have been obtained independently by K. Wagner [25]; Theorem 1 and many of the other results of section 2 were independently obtained by Wagner. He also has work closely related to our work in section 4.

2 Truth-Table Reducibility to SAT

Polynomial time truth-table reducibility ($\leq^p_{tt}$) was first introduced by Ladner, Lynch and Selman [21] and further studied by Ladner and Lynch [20]. The general idea is that $A \leq^p_{tt} B$ 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 Wagner [24] and Köbler, Schöning and Wagner [18].)

2.1 Some Definitions and a Main Theorem

Definition Let A and B be decision problems. Let $\text{eval}(z, x_1, \ldots, x_n)$ 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) $A \leq^p_{tt} B$, 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) $A \leq^p_{bf} B$, 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) $A \leq^p_{ftt} B$, 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.

Definition For fixed k, three notions of k-truth-table reducibility, $A \leq^p_{k\text{-}tt} B$, $A \leq^p_{k\text{-}bf} B$ and $A \leq^p_{k\text{-}ftt} B$ are defined exactly as above except that now $k > 0$ is a constant.

Definition Let A be a decision problem. Then $\leq^p_{tt}(A)$, $\leq^p_{bf}(A)$ and $\leq^p_{ftt}(A)$ 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, $\leq^p_{tt}(\text{NP})$, $\leq^p_{bf}(\text{NP})$ and $\leq^p_{ftt}(\text{NP})$ denote $\leq^p_{tt}(\text{SAT})$, $\leq^p_{bf}(\text{SAT})$ and $\leq^p_{ftt}(\text{SAT})$, respectively.

It is obvious that $\leq^p_{tt}(\text{NP}) \supseteq \leq^p_{bf}(\text{NP}) \supseteq \leq^p_{ftt}(\text{NP})$; complete problems for these classes are given by Köbler, Schöning and Wagner [18] and in Wagner [24]. The next theorem states that $\leq^p_{tt}(\text{NP})$ is actually equal to $\leq^p_{bf}(\text{NP})$. (It is still open whether $\leq^p_{tt}(A) = \leq^p_{bf}(A)$ for arbitrary A.)

Theorem 1 $\leq^p_{tt}(\text{NP}) = \leq^p_{bf}(\text{NP})$.

Proof Let Σ be a finite alphabet and suppose $C \subseteq \Sigma^*$ and $C \leq^p_{tt} \text{SAT}$; 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 $\text{SAT}(y)$ is the predicate "y is satisfiable", also let $\text{TRU}(x, y)$ be the polynomial time predicate "x codes a satisfying assignment of y"; so $\text{SAT}(y) \leftrightarrow (\exists x < y)\text{TRU}(x, y)$. Let $B(i, x)$ be the predicate

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

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

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

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 $j \le i$. 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

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

Finally we claim that this puts C in $\le^p_{bf}(\mathrm{SAT})$. 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 function 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 $\le^p_{tt}(\mathrm{NP})$ if and only if it is logspace truth-table reducible to* SAT *via Boolean formulas.*

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

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

Proof This is immediate from the theorem of Ladner and Lynch [20] that log space Turing reducibility is identical to a form of log space truth-table reducibility. Actually Ladner and Lynch's notion of log space truth-table reducibility is slightly different from ours, but nonetheless is intermediate in strength between log space truth-table reducibility via Boolean formulas and log space truth-table reducibility via Boolean circuits; thus it follows from Theorems 1 and 2 that all these reducibilities are equivalent for reductions to SAT. □

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

Next we discuss a syntactic characterization of formulas which define predicates in $\leq_{tt}^{p}(\mathrm{NP})$. We show 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 [4]. 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 $\Sigma_2^p \cap \Pi_2^p = \leq_{tt}^{p}(\mathrm{NP})$.)

Definition 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 length of $\vec{x}$.

Definition An NP-*formula* is a formula which defines a predicate in NP (alternatively, let an NP-formula be a Σ_1^b-formula of [4]). $\Sigma_2^b \cap \Pi_2^b$ 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 4 *The class* $\leq_{tt}^{p}(\mathrm{NP})$ *is equal to the class of predicates definable by* $\Sigma_2^b \cap \Pi_2^b$ *formulas.*

Proof The proof of Theorem 1 showed that every predicate in $\leq_{tt}^{p}(\mathrm{NP})$ 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 $\Sigma_2^b \cap \Pi_2^b$ formula. For the converse, note 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 each Q_i is $\forall$ or $\exists$, 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. $\Box$

Corollary 5 *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*-formulas 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. $\Box$

Cai and Hemachandra [5] define a hierarchy of sets denoted Σ_k^L, Π_k^L for $k \geq 0$. They define these sets in terms of Turing machine computation; we give an equivalent syntactic definition:

Definition A predicate $A \subseteq \mathbb{N}^r$ is in Σ_k^L if and only if A can be defined as the set of $\vec{x} \in \mathbb{N}^r$ such that

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

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 Corollary 5, 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

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

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. □

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

Krentel [19] 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 $\mathrm{P}^{\mathrm{SAT}}(O(\log n)) = \leq_{tt}^p(\mathrm{NP})$.

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.

$\supseteq$: (This is based on a method used by Beigel [2].) 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

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

where each predicate is implied by the next. Of course $B(k(x)+2,x)$ 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 with $O(\log n)$ queries in polynomial time. $\square$

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 is explored more fully in section §4 below.

To summarize, we list most of the characterizations of $\leq^p_{tt}(\mathrm{NP})$ discussed above:

Theorem 8 *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 log space, Boolean formula truth-table reducible to* SAT*,*
- *A is log space, Turing reducible to* SAT*,*
- *A is definable by a $\Sigma^b_2 \cap \Pi^b_2$ formula,*
- *A is in Σ^L_2 (equivalently, Π^L_2),*
- *A is in $\mathrm{P}^{\mathrm{SAT}}(O(\log n))$.*

See Theorem 15 below for two more equivalent definitions of $\leq^p_{tt}(\mathrm{NP})$. See also the next section for a related result.

3 Multiple Rounds of Parallel Queries

We may characterize $\leq^p_{tt}(\mathrm{NP})$ as containing 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 $C(x)$ 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 $\leq^p_{tt}(\mathrm{NP})$, 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". We order the k-tuples lexicographically so $\vec{r} \prec \vec{s}$ iff for some i, $r_i < s_i$ and $(\forall j < i)(r_j = s_j)$. Here $\vec{r} = \langle r_1, \ldots, r_k \rangle$ is a sequence of integers.

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:

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

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 $w_j \leq 2^{p(n)}$. 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 length of $\vec{s}$ is bounded by a polynomial in n. Accordingly the $\exists \vec{s}$ quantifier is sharply bounded. Thus by Theorem 4, C is in $\leq^p_{tt}(\mathrm{NP})$. (We could avoid the use of Theorem 4 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^p_2, 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, $\sum \log(f_j(x) + 1) = O(\log n)$ then M recognizes a predicate in $\leq^p_{tt}(\mathrm{NP})$. This fact has the $\subseteq$-inclusion of Theorem 7 as a special case with $k = O(\log n)$ and $f_j = O(1)$.

4 The Difference Hierarchy

The difference (or Boolean) hierarchy has had a long history in the enterprise of classifying the complexity of sets. It was first introduced in the context of descriptive set theory by Hausdorff [14] and was studied in a general setting for applications to logic by Addison [1]. It was developed in the context of recursion theory by Ersov [9] and proved useful in the classification of index sets [9, 15, 17]. The connections between the levels of the finite difference hierarchy over the r.e. sets and bounded truth-table reducibility to the halting problem was noted in [13] and further explored in [16] and [3]. The difference hierarchy over the r.e. sets was extended to the transfinite by Eršov [10], who noted the connection between this hierarchy and the 'bounded mind changes' characterization of Δ_2^0 observed by [11] and [23] and further explored in [7] and [8].

Differences of NP sets were first considered by Papadimitriou and Yannakakis [22] where the notion again proved useful for the classification of natural problems. The difference hierarchy was extended through the finite levels by Cai and Hemachandra [5] and Wagner and Wechsung [26], who noted the connections with bounded truth-table reducibility to SAT, and was further studied by Beigel [2]. In this section we consider the infinite difference hierarchy over NP and investigate the connection with truth-table reducibility and Turing reducibility to SAT. We shall see that Δ_2^p is in some sense an "infinitary" version of the Boolean hierarchy over NP in that a set in Δ_2^p can be expressed as the differences of exponentially many sets in NP.

4.1 The infinite difference hierarchy over NP

Let $(S, \leq_S)$ be a linearly ordered set of order type ω (i.e. order-isomorphic to the natural number $\mathbb{N}$). Call $x \in S$ *even* if it has an even number of $\leq_S$ predecessors.

Definition The structure $(S, \leq_S)$ is an *ω-order with polynomial parity* (abbreviated p.p.o) if there is a polynomial time parity function f_S such that for all $x \in S$, $f_S(x) = 0$ if x is even and $f_S(x) = 1$ otherwise.

The most obvious examples of p.p.o.'s are natural numbers under the usual order and $\{0,1\}^*$ under the lexicographical order.

Definition Let $(S, \leq_S)$ be a p.p.o. and let $\{R_x\}_{x \in S}$ be an indexed sequence of sets.

(1) $\{R_x\}_{x \in S}$ is *uniformly NP* if the two-place predicate "$\sigma \in R_x$" is in NP;

(2) $\{R_x\}_{x\in S}$ is *NP*$\supseteq$ (resp. *NP*$\subseteq$) if it is uniformly NP and, in addition, $x \leq_S y$ implies $R_x \supseteq R_y$ (resp., $R_x \subseteq R_y$).

Definition Let S be a p.p.o., $S = \{s_0 < s_1 < s_2 < \cdots\}$

(1) If $\mathbb{R} = \{R_x\}_{x\in S}$ is *NP*$\supseteq$, then

$$\begin{aligned}\Sigma(\mathbb{R}) &= \Big\{\sigma \in R_{s_0} - \bigcap_{x\in S} R_x : (\max_S x)(\sigma \in R_x) \text{ is even}\Big\} \\ &= \bigcup_{i\geq 0}\big(R_{s_{2i}} - R_{s_{2i+1}}\big)\,;\end{aligned}$$

(2) If $\mathbb{R} = \{R_x\}_{x\in S}$ is *NP*$\subseteq$, then

$$\begin{aligned}\Sigma(\mathbb{R}) &= \Big\{\sigma \in \bigcup_{x\in S} R_x : (\min_S x)(\sigma \in R_x) \text{ is odd}\Big\} \\ &= \bigcup_{i\geq 0}\big(R_{s_{2i+1}} - R_{s_{2i}}\big)\,;\end{aligned}$$

So $\Sigma(\mathbb{R})$ is just the set defined taking infinite differences of the R_x's. If $\mathbb{R}$ is *NP*$\supseteq$ (resp. *NP*$\subseteq$) let $h(\sigma)$ be the maximum (resp. minimum) x such that $\sigma \in R_x$. Then $h(\sigma)$ is defined and even iff $\sigma \in \Sigma(\mathbb{R})$. The case that is of special interest to us is when $h(\sigma)$ is polynomially bounded; i.e., $|h(\sigma)| \leq k \cdot |\sigma|^k$ for some constant k whenever $h(\sigma)$ is defined. In this case h is an FP^{SAT} function as h can be computed using a binary search with queries to SAT, exploiting the fact that $\mathbb{R}$ is descending (resp. ascending). (Recall that Krentel [19] defines a function f to be in FP^{SAT} if it is computable in polynomial time by a transducer with an oracle for SAT.)

Definition $\mathrm{NP}_T^{\supseteq}(\omega)$ is the class of sets A for which there is a p.p.o. S, an *NP*$\supseteq$ sequence $\mathbb{R} = \{R_x\}_{x\in S}$ with $\bigcap R_x = \emptyset$, and a function $h \in FP^{\mathrm{SAT}}$ such that $A = \Sigma(\mathbb{R})$ and

$$\begin{aligned}&\sigma \notin \textstyle\bigcup_{x\in S} R_x \rightarrow h(\sigma) = \# \notin S \\ &\sigma \in \textstyle\bigcup_{x\in S} R_x \rightarrow h(\sigma) = \max_S\{x : \sigma \in R_x\}\end{aligned}$$

Here $\#$ is a special symbol not in S.

Likewise, $\mathrm{NP}^{\subseteq}_{T}(\omega)$ is the class of sets A for which there is a p.p.o. S, an *NP*$\subseteq$ sequence $\mathbb{R} = \{R_x\}_{x \in S}$ with $\bigcap R_x = \emptyset$, and a function $h \in FP^{\mathrm{SAT}}$ such that $A = \Sigma(\mathbb{R})$ and

$$\begin{aligned} &\sigma \notin \textstyle\bigcup_{x \in S} R_x \rightarrow h(\sigma) = \# \notin S \\ &\sigma \in \textstyle\bigcup_{x \in S} R_x \rightarrow h(\sigma) = \min_S\{x : \sigma \in R_x\} \end{aligned}$$

A truth assignment to $x_1, ..., x_n$ is identified with a string of 0's and 1's of length n; the i-th symbol is a 1 iff the variable x_i is assigned the truth value *True*.

Lemma 10 *Let* $\{0,1\}^*$ *be ordered lexicographically and define*

$$\begin{aligned} \text{ODDMAXSAT} = \{\phi(x_1, \ldots, x_n) : {} & \phi \in \text{SAT and} \\ & x_n = 1 \text{ in } \phi\text{'s maximum satisfying assignment}\}, \\ \text{EVENMINSAT} = \{\phi(x_1, \ldots, x_n) : {} & \phi \in \text{SAT and} \\ & x_n = 0 \text{ in } \phi\text{'s minimum satisfying assignment}\}, \end{aligned}$$

then

(a) ODDMAXSAT *is in* $\mathrm{NP}^{\supseteq}_{T}(\omega)$*, and*

(b) EVENMINSAT *is in* $\mathrm{NP}^{\subseteq}_{T}(\omega)$*.*

Proof (a) Let $S = \{0,1\}^*$ be ordered lexicographically (where the initial element is the empty string ϵ), and let # be a new symbol. For $x \in S$, define

$$R_x = \{\phi(x_1, \ldots, x_n) : (\exists y \geq_S x) y \text{ satisfies } x\}.$$

The sequence $\mathbb{R} = \{R_x\}_{x \in S}$ is uniformly NP, since given ϕ and x one can guess a satisfying assignment y for ϕ, then in polynomial time check whether $y \geq x$. $\mathbb{R}$ is clearly a *NP*$\supseteq$-sequence, and $\bigcap_{x \in S} R_x = \emptyset$. Now

$$\begin{aligned} \phi(x_1, \ldots, x_n) \in \Sigma(\mathbb{R}) &\leftrightarrow \sigma \in \textstyle\bigcup_{x \in S} R_x \text{ and } (\max_S x)(y \in R_x) \text{ is even} \\ &\leftrightarrow \phi(x_1, \ldots, x_n) \in \text{ODDMAXSAT}. \end{aligned}$$

To compute $h(\phi)$, first ask if $\phi \in \mathrm{SAT}$. If not, output #. If yes, the maximum satisfying assignment y of ϕ can found in polynomial time by binary search using SAT as an oracle. Thus $h(\phi) \in FP^{\mathrm{SAT}}$.

(b) is proved similarly using $S = \{0,1\}^*$, ordered lexicographically, and $R_x = \{\phi(x_1, \ldots, x_n) : (\exists y <_S x) y \text{ satisfies } \phi\}$. $\square$

We now consider the relation between Δ_2^p and the infinite difference hierarchy.

Theorem 11 *The following are equivalent:*

(i) A is in Δ_2^p;

(ii) A is in $\mathrm{NP}_T^{\supseteq}(\omega)$;

(iii) A is in $\mathrm{NP}_T^{\subseteq}(\omega)$;

Proof *(ii)* implies *(i)*: Assume there is a p.p.o S with parity function $f_S \in \mathrm{P}$, an *NP*$\supseteq$ sequence $\mathbb{R} = \{R_x\}_{x \in S}$ and h in FP^{SAT} such that $A = \Sigma(\mathbb{R})$ and $\sigma \in \bigcup_{x\in S} R_x \rightarrow h(\sigma) = \max_S\{x : \sigma \in R_x\}$. To decide if $\sigma \in A$, compute $h(\sigma)$; if equal to $\#$, output 'no'. If $h(\sigma) = x \in S$, $\sigma \in A \leftrightarrow f_S(x) = 0$, hence A is in $\mathrm{P}^{\mathrm{SAT}} = \Delta_2^P$. *(iii)* implies *(i)* is shown by a similar argument.

To show that *(i)* implies *(ii)*, we use the fact that ODDMAXSAT is m-complete for Δ_2^p. This fact is due to Krentel [19]; by *m-complete* we mean complete under polynomial time many-one reductions. Hence there exists $g \in \mathrm{P}$ such that $\sigma \in A \leftrightarrow g(\sigma) \in \mathrm{ODDMAXSAT}$. Let S, $\mathbb{R}$ and h be as in the proof of Lemma 10(a), and define $R'_x = g^{-1}(R_x)$ for all $x \in S$. Then $\mathbb{R}' = \{R'_x\}_{x\in S}$ is *NP*$\supseteq$, and

$$\begin{aligned} \sigma \in A \leftrightarrow & g(\sigma) \in \mathrm{ODDMAXSAT} = \Sigma(\mathbb{R}) \\ \leftrightarrow & g(\sigma) \in \textstyle\bigcup_{x\in S} R_x \text{ and } (\max_S x)(g(\sigma) \in R_x) \text{ is even} \\ \leftrightarrow & \sigma \in \textstyle\bigcup_{x\in S} R'_x \text{ and } (\max_S x)(\sigma \in R'_x) \text{ is even.} \end{aligned}$$

So $A = \Sigma(\mathbb{R})$ while $h' = h \circ g$ computes the maximum x such that $\sigma \in R'_x$ if such an x exists.

That *(i)* implies *(iii)* follows in a similar fashion from the fact that EVENMINSAT is also m-complete for Δ_2^p. $\square$

We next consider the relationship between the infinite difference hierarchy and the class $\leq_{tt}^p(\mathrm{NP})$.

Lemma 12 *A set C is in $\leq_{tt}^p(\mathrm{NP})$ if and only if there is a polynomial time function $f(x) = \langle y_1, \ldots, y_{n(x)} \rangle$ such that $\chi_{\mathrm{SAT}}(y_1) \geq \chi_{\mathrm{SAT}}(y_2) \geq \cdots \geq \chi_{\mathrm{SAT}}(y_{n(x)})$ and*

$$\begin{aligned} x \in C \leftrightarrow & |\{j : y_j \in \mathrm{SAT}\}| \text{ is odd} \\ \leftrightarrow & (\max j)(y_j \in \mathrm{SAT}) \text{ is odd.} \end{aligned}$$

Proof Suppose C is in $\leq_{tt}^{p}(\mathrm{NP})$. As shown above in the proof of Theorem 7, there is a polynomial time function $k(x)$ and NP predicates $A^*(i,x)$ and $B(i,x)$ such that for each i, $B(i+1,x)$ implies $A^*(i,x)$ which in turn implies $B(i,x)$, and

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

Now there are polynomial time functions f_{A^*} and f_B such that

$$\begin{aligned} A^*(i,x) &\leftrightarrow f_{A^*}(i,x) \in \mathrm{SAT} \\ B(i,x) &\leftrightarrow f_B(i,x) \in \mathrm{SAT} \end{aligned}$$

If we set $n(x) = 2k(x)+2$ and define

$$\begin{aligned} y_{2i+1} &= f_{A^*}(i,x) \\ y_{2i} &= f_B(i,x) \end{aligned}$$

then $\chi_{\mathrm{SAT}}(y_1) \geq \chi_{\mathrm{SAT}}(y_2) \geq \cdots \chi_{\mathrm{SAT}}(y_{n(x)})$ and

$$x \in C \leftrightarrow \bigvee_{i=1}^{k(x)+1} (y_{2i-1} \in \mathrm{SAT} \wedge y_{2i} \notin \mathrm{SAT})$$

It follows that

$$\begin{aligned} x \in C &\leftrightarrow (\max j)(y_j \in \mathrm{SAT}) \text{ is odd} \\ &\leftrightarrow |\{j : y_j \in \mathrm{SAT}\}| \text{ is odd.} \qquad \square \end{aligned}$$

Definition Let

$$\begin{aligned} \mathrm{PARITY}_{\omega}^{\mathrm{SAT}} &= \bigcup_n \Big\{ \langle \phi_0, \ldots, \phi_{n-1} \rangle : |\{j : \phi_j \in \mathrm{SAT}\}| \text{ is odd} \Big\} \\ \mathrm{DECPAR}_{\omega}^{\mathrm{SAT}} &= \bigcup_n \Big\{ \langle \phi_0, \ldots, \phi_{n-1} \rangle : \forall j (\chi_{\mathrm{SAT}}(\phi_j) \geq \chi_{\mathrm{SAT}}(\phi_{j+1}) \\ &\qquad \text{and } (\max j)(\chi_{\mathrm{SAT}}(\phi_j) = 1) \text{ is even} \Big\} \\ \mathrm{INCPAR}_{\omega}^{\mathrm{SAT}} &= \bigcup_n \Big\{ \langle \phi_0, \ldots, \phi_{n-1} \rangle : \forall j (\chi_{\mathrm{SAT}}(\phi_j) \leq \chi_{\mathrm{SAT}}(\phi_{j+1}) \\ &\qquad \text{and } (\min j)(\chi_{\mathrm{SAT}}(\phi_j) = 1) \text{ is even} \Big\} \end{aligned}$$

Theorem 13

(a) $\mathrm{PARITY}_\omega^{\mathrm{SAT}}$ *is m-complete for* $\leq_{tt}^p(\mathrm{NP})$,

(b) $\mathrm{DECPAR}_\omega^{\mathrm{SAT}}$ *is m-complete for* $\leq_{tt}^p(\mathrm{NP})$,

(c) $\mathrm{INCPAR}_\omega^{\mathrm{SAT}}$ *is m-complete for* $\leq_{tt}^p(\mathrm{NP})$.

Proof $\mathrm{PARITY}_\omega^{\mathrm{SAT}}$, $\mathrm{DECPAR}_\omega^{\mathrm{SAT}}$ and $\mathrm{INCPAR}_\omega^{\mathrm{SAT}}$ are all evidently in $\leq_{tt}^p(\mathrm{NP})$; hence (a) and (b) follow readily from Lemma 12. To prove (c), note that $\mathrm{DECPAR}_\omega^{\mathrm{SAT}} \leq_m^p \mathrm{INCPAR}_\omega^{\mathrm{SAT}}$ via the following reduction. Let ϕ_n be an unsatisfiable formula and let $n' = 2\lfloor\frac{1}{2}n\rfloor$. If we let $\phi'_i = \phi_{n'-i}$ it is easily verified that $\langle\phi_0, \ldots \phi_{n-1}\rangle$ is in $\mathrm{DECPAR}_\omega^{\mathrm{SAT}}$ if and only if $\langle\phi'_0, \ldots \phi'_{n'}\rangle$ is in $\mathrm{INCPAR}_\omega^{\mathrm{SAT}}$. $\square$

Next we consider an additional restriction on $\mathrm{NP}_T^{\supseteq}(\omega)$ and $\mathrm{NP}_T^{\subseteq}(\omega)$ sets; namely, we assume there is a polynomial time function $\overline{h}(\sigma)$ which computes a list of indices from S such that the maximum (resp. minimum) value $x = h(\sigma)$ satisfying $\sigma \in R_x$ is in the list $\overline{h}(x)$. In Theorem 15 we see that this characterizes sets in $\leq_{tt}^p(\mathrm{NP})$.

Definition $\mathrm{NP}_{tt}^{\supseteq}(\omega)$ is the class of sets A for which there is a p.p.o. S with parity function f_S, an $NP\supseteq$ sequence $\mathbb{R} = \{R_x\}_{x\in S}$ with $\bigcap_{x\in S} R_x = \emptyset$ and $A = \Sigma(\mathbb{R})$, and a function $\overline{h} \in \mathrm{P}$ such that for all σ, $\overline{h}(\sigma) = \left\langle y_0, \ldots, y_{n(\sigma)}\right\rangle$ with $y_0 <_S y_1 <_S \cdots <_S y_n$ and y_i S-even iff i is even and such that for $\sigma \in A$, $(\max_S x)(\sigma \in R_x)$ is in $\overline{h}(\sigma)$.

$\mathrm{NP}_{tt}^{\subseteq}(\omega)$ is the class of sets A for which there is a p.p.o. S with parity function f_S, an $NP\subseteq$ sequence $\mathbb{R} = \{R_x\}_{x\in S}$ with $A = \Sigma(\mathbb{R})$, and a function $\overline{h} \in \mathrm{P}$ such that for all σ, $\overline{h}(\sigma) = \left\langle y_0, \ldots, y_{n(\sigma)}\right\rangle$ with $y_0 >_S y_1 >_S \cdots >_S y_n$ and y_i S-even iff i is even and such that for $\sigma \in A$, $(\min_S x)(\sigma \in R_x)$ is in $\overline{h}(\sigma)$.

Lemma 14

(a) $\mathrm{DECPAR}_\omega^{\mathrm{SAT}}$ *is in* $\mathrm{NP}_{tt}^{\supseteq}(\omega)$;

(b) $\mathrm{INCPAR}_\omega^{\mathrm{SAT}}$ *is in* $\mathrm{NP}_{tt}^{\subseteq}(\omega)$.

Proof (a) Let $S = \mathbb{N}$ under the usual order and define

$$R_x = \{\langle\phi_0, \ldots, \phi_{n-1}\rangle : x < n \wedge (\forall j \leq x)\phi_j \in \mathrm{SAT}\}.$$

then $\mathbb{R} = \{R_x\}_{x \in S}$ is in $NP\supseteq$, $\bigcap_{x\in S} R_x = \emptyset$ and $\Sigma(\mathbb{R}) = \mathrm{DECPAR}^{\mathrm{SAT}}_{\omega}$. If $\sigma = \langle \phi_0, \ldots, \phi_{n-1}\rangle$ and we set $\overline{h}(\sigma) = \langle 0, 1, \ldots, n-1\rangle$ then $\overline{h}$ is clearly polynomial time computable and

$$\sigma \in \mathrm{DECPAR}^{\mathrm{SAT}}_{\omega} \to (\max x)(\sigma \in R_x) \in \overline{h}(\sigma).$$

(b) is proved similarly, defining $R_0 = \emptyset$ and

$$R_{x+1} = \{\langle \phi_0, \ldots, \phi_{n-1}\rangle : (\forall j \geq x)(j < n \to \phi_i \in \mathrm{SAT})\}. \qquad \Box$$

Theorem 15 *The following are equivalent:*

(i) A is in $\leq^p_{tt}(\mathrm{NP})$,

(ii) A is in $\mathrm{NP}^{\supseteq}_{tt}(\omega)$,

(iii) A is in $\mathrm{NP}^{\subseteq}_{tt}(\omega)$,

Proof *(ii)* implies *(i)*: Assume A is in $\mathrm{NP}^{\supseteq}_{tt}(\omega)$ and let S, $\mathbb{R} = \{R_x\}$ and $\overline{h}$ be as in the definition. Since $\{R_x\}_{x\in S}$ is (uniformly) $NP\supseteq$, there is a polynomial time function $g(x, \sigma)$ such that $\sigma \in R_x \leftrightarrow g(x, \sigma) \in \mathrm{SAT}$. Then $A \leq^p_{tt} \mathrm{SAT}$ via the polynomial time function which maps σ to the Boolean formula

$$\bigvee_{k=0}^{\lfloor \frac{1}{2} n(\sigma) \rfloor} (g(y_{2k}, \sigma) \in \mathrm{SAT} \wedge g(y_{2k+1}, \sigma) \notin \mathrm{SAT})$$

A similar argument shows *(iii)* implies *(i)*.

The proof that *(i)* implies *(ii)* and *(iii)* proceeds just like that of Theorem 11 using Theorem 13 and Lemma 14. $\Box$

Next we exhibit some new natural examples of sets complete at the finite and infinite levels of the difference hierarchy over NP. Following Cai and Hemachandra [6] we use NP(n) to denote the n-th level of the difference hierarchy over NP.

Definition Let $V(\phi) = \{i : \phi$ has a satisfying assignment setting $x_i = 1\}$. For each n, let $V_n(\phi) = V(\phi) \cap \{0, 1, \ldots, n-1\}$ and define

$$A_n = \{\phi : |V_n(\phi)| \text{ is odd}\},$$

$$B_n = \{\phi : (\max i)(i \in V_n(\phi)) \text{ is even}\},$$

$$A_\omega = \{\phi : |V(\phi)| \text{ is odd}\},$$

$$B_\omega = \{\phi : (\max i)(i \in V(\phi)) \text{ is even}\}.$$

Theorem 16

(a) *For all n, A_n and B_n are m-complete for* $\mathrm{NP}(n)$*,*

(b) *A_ω and B_ω are m-complete for* $\mathrm{NP}^{\supset}_{tt}(\omega)$ *(* $= \leq^p_{tt}(\mathrm{NP})$ *).*

Proof For each n, A_n and B_n are evidently in $\mathrm{NP}(n)$. A_ω is seen to be in $\mathrm{NP}^{\supset}_{tt}(\omega)$ by letting S be the positive integers, R_x be $\{\phi : |V(\phi)| \geq x\}$ and $\overline{h}(\phi(x_0, \ldots, x_{n-1})) = \langle 1, \ldots, n\rangle$. B_ω is seen to be in $\mathrm{NP}^{\supset}_{tt}(\omega)$ by letting S be the natural numbers, R_x be $\{\phi : (\exists i \geq x)(i \in V(\phi)\}$ and $\overline{h}(\phi(x_0, \ldots, x_{n-1})) = \langle 0, \ldots, n-1\rangle$. For each n let $\mathrm{PARITY}^{\mathrm{SAT}}_n$ be $\{(\phi_0, \ldots, \phi_{n-1}) : |\{j : \phi_j \in \mathrm{SAT}\}| \text{ is odd}\}$. By Observation 3.4(i) of Beigel [2], $\mathrm{PARITY}^{\mathrm{SAT}}_n$ is m-complete for $\mathrm{NP}(n)$. That A_n is m-complete for $\mathrm{NP}(n)$ follows from the fact that $\mathrm{PARITY}^{\mathrm{SAT}}_n$ is many-one reducible to it via the the following reduction: Given $\phi_0, \ldots, \phi_{n-1}$ let ϕ'_i be obtained from ϕ_i by replacing the variables $x_0, \ldots, x_{n-1}$ by new variables and defining

$$\phi^* = \bigvee_{0 \leq i < n} \left(\phi'_i \wedge x_i \wedge \bigwedge_{0 \leq j < i} \overline{x}_j \wedge \bigwedge_{i < j < n} \overline{x}_j \right).$$

Then $\langle \phi_0, \ldots, \phi_{n-1}\rangle \in \mathrm{PARITY}^{\mathrm{SAT}}_n$ iff $\phi^* \in A_n$. To see that B_n is m-complete for $\mathrm{NP}(n)$, let

$$C_n = \{ \langle \phi_0, \ldots, \phi_{n-1}\rangle : \chi_{\mathrm{SAT}}(\phi_j) \geq \chi_{\mathrm{SAT}}(\phi_{j+1}) \text{ for each } j \text{ and } (\max i)(\phi_i \in \mathrm{SAT}) \text{ is even}\}.$$

C_n is easily seen to be m-complete for $\mathrm{NP}(n)$ using the normal form for sets in $\mathrm{NP}(n)$; namely, every set A in $\mathrm{NP}(n)$ can be expressed as

$$A = \bigcup_{0 \leq i \leq \lfloor \frac{1}{2} n - 1 \rfloor} (L_{2i} - L_{2i+1})$$

where the L_i's are a descending sequence of NP sets with $L_n = \emptyset$. Thus,

$$\begin{aligned} \sigma \in A &\leftrightarrow (\max i)(\sigma \in L_i) \text{ is even} \\ &\leftrightarrow (\max i)(f(\sigma, i) \in \mathrm{SAT}) \text{ is even} \end{aligned}$$

where $\phi_i = f(\sigma, i)$ is a reduction of L_i to SAT and if ϕ_{i+1} is in SAT then so is ϕ_i. It follows that B_n is m-complete for $\mathrm{NP}(n)$ since the reduction $\langle \phi_0, \ldots \phi_{n-1}\rangle \mapsto \phi^*$ given above shows that $C_n \leq^p_m B_n$.

The same reductions give $\mathrm{PARITY}_\omega^{\mathrm{SAT}} \leq_m^p A_\omega$ and $\mathrm{DECPAR}_\omega^{\mathrm{SAT}} \leq_m^p B_\omega$. It follows from Theorem 13 that A_ω and B_ω are m-complete for $\leq_{tt}^p(\mathrm{NP})$ and hence for $\mathrm{NP}_{tt}^{\supset}(\omega)$ by Theorem 15. $\square$

5 Separation Results

Cai and Hemachandra [6] showed that there are oracles relative to which the finite difference hierarchy has infinitely many levels, and oracles relative to which the difference hierarchy extends exactly k levels, with PSPACE collapsing to the $(k+1)$-st level. We extend these results to level ω by showing that even in relativized worlds where the difference hierarchy 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.

Theorem 17 *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 m-complete for NP^A. To obtain a set A with $\mathrm{PSPACE}^A \leq_{tt}^p S^A$, we can modify the construction in [6] of an oracle A which separates the finite levels of the difference hierarchy as follows: spread out the "blocks" used in that construction and intersperse a coding of a set U which is PSPACE-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. □

Following Goldsmith and Joseph [12], define a *relativized Boolean formula* to be of the form

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

where ϕ is in conjunctive normal form and each σ_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 $\overline{x}_j$. 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 the σ_i's and τ_i's as strings in $\{0,1\}^*$ with 1 for "True" and 0 for "False". Now define SAT^A to be the set of satisfiable relativized Boolean formulas. It was shown by Goldsmith and Joseph that SAT^A is m-complete for NP^A for all A. Let $\mathrm{ODDMAXSAT}^A$ be the set of satisfiable relativized Boolean formula whose maximum satisfying assignment is odd; $\mathrm{ODDMAXSAT}^A$ is evidently in $\leq_T^p(\mathrm{NP}^A)$.

Theorem 18 *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 describe a diagonalization that ensures that $\mathrm{ODDMAXSAT}^A$ is in $\leq_T^p(\mathrm{NP}^A)$ but not in $\leq_{tt}^p(\mathrm{NP}^A)$. The steps 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, $f_i(x) = \langle y_1, \ldots, y_{n(x)} \rangle$ where each y_j encodes a relativized Boolean formula. By the relativized form of Lemma 12, a set B is $\leq_{tt}^p(\mathrm{SAT}^A)$ if and only if there is an i such that for each x, $f_i(x) = \langle y_1, \ldots, y_{n(x)} \rangle$ where $\chi_{\mathrm{SAT}^A}(y_1) \geq \chi_{\mathrm{SAT}^A}(y_2) \geq \cdots \geq \chi_{\mathrm{SAT}^A}(y_{n(x)})$ and

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

We call such an f_i a *normalized* truth table reduction of B to SAT^A. We shall construct an oracle A for which there is no 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. At stage s there will be a set S of strings available for addition to A. We will maintain the condition that $|S| > 2^{n-1}$ so there will always be both even and odd strings available. 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. If $f_s(\Phi_n) = \langle y_1, \ldots, y_k \rangle$, let $h_A(\Phi_n) = \left|\{j : y_j \in \mathrm{SAT}^A\}\right| \bmod 2$. We will try 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 $n = n_{s+1} > n_s$ 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 $\phi_j \wedge \bigwedge \sigma_i \in A \wedge \bigwedge \tau_i \notin A$. 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 $\tau_i \notin A$. Let X be the set $\{j : y_j \in \mathrm{SAT}^{A_s}\}$ and let $h_s(\Phi_n) = (|X| \bmod 2)$. If $h_s(\Phi_n) = 1$ then we may set $A_{s+1} = A_s$; in this case, $h_A(\Phi_n) = h_s(\Phi_n) = 1$ 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. 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 satisfying assignment of y_j with respect to the oracle A_{s+1} constructed so far. and let N_j be the corresponding set of strings specified 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 $S' := \{\sigma \in S : \sigma(x_n) = 1 \leftrightarrow h_s(\Phi_n) = 0\}$. S' is the set of strings such that if we can add a member of S' to A_{s+1} without altering the satisfiability of the members of $f_s(\Phi_n)$ (i.e., without altering $h_s(\Phi_n)$) 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 $\Phi_n \in \mathrm{ODDMAXSAT}^{A_{s+1}}$ iff $h_s(\Phi_n) = 0$. 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

$\{j : y_j \in \mathrm{SAT}^{A_{s+1}}\}$ and redefine $h_s(\Phi_n) = |X| \bmod 2$. 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. 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 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. $\Box$

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 $\mathrm{ODDMAXSAT}^A \leq_{tt}^{\mathrm{P}^A} SAT^A$ and thus to ensure that $\leq_{tt}^{\mathrm{P}^A} (\mathrm{NP}^A) \neq \leq_X^p(\mathrm{NP}^A)$.

References

[1] J. W. Addison, *The method of alternating chains*, in Symposium on the Theory of Models, North-Holland, 1965, pp. 1–15.

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

[3] R. Beigel, W. I. Gasarch, and L. Hay, *Bounded query classes and the difference hierarchy*, Tech. Rep. TR-1847, Dept. of Computer Science, Univ. of Maryland at College Park, 1987.

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

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

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

[7] H. G. Carstens, *Δ_2^0-mengen*, Archiv für Mathematische Logik und Grundlagenforschung, 18 (1976), pp. 55–65.

[8] R. L. Epstein, R. Haas, and R. L. Kramer, *Hierarchies of sets and degrees below* $0'$, in Logic Year 1979-80, Lecture Notes in Mathematics #859, Springer-Verlag, 1981, pp. 32–48.

[9] Y. L. Eršov, *A hierarchy of sets I*, Algebra and Logic, 7 (1968), pp. 25–43.

[10] ——, *A hierarchy of sets II*, Algebra and Logic, 7 (1968), pp. 212–232.

[11] E. M. Gold, *Limiting recusion*, Journal of Symbolic Logic, 30 (1965), pp. 28–48.

[12] 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.

[13] J. Hartley Rogers, *Theory of Recursive Functions and Effective Computability*, McGraw-Hill, 1967. Reprinted by MIT Press, 1987.

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

[15] L. Hay, *Index sets universal for differences of arithmetic sets*, Zeitschrift für Mathematische Logik und Grundlagen der Mathematik, 20 (1974), pp. 239–254.

[16] ——, *Convex subsets of 2^n and bounded truth table reducibility*, Discrete Mathematics, 21 (1978), pp. 31–46.

[17] L. Hay, A. B. Manaster, and J. G. Rosenstein, *Small recursive ordinals, many-one degress and the arithmetical difference hierarchy*, Annals of Mathematical Logic, 8 (1975), pp. 297–343.

[18] 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.

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

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

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

[22] C. H. Papadimitiou and M. Yannakakis, *The complexity of facets (and some facets of complexity)*, J. Comput. System Sci., 28 (1984), pp. 244–259.

[23] H. Putnam, *Trial and error predicates and solution of a problem of Mostowski*, Journal of Symbolic Logic, 30 (1965), pp. 49–57.

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

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

[26] 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.