TIME-SPACE TRADEOFFS FOR COUNTING NP SOLUTIONS MODULO INTEGERS

R. Ryan Williams

Abstract.
We prove the first time-space tradeoffs for counting the number of solutions to an NP problem modulo small integers, and also improve upon known time-space tradeoffs for SAT. Let $m > 0$ be an integer, and define MOD_m-SAT to be the problem of determining if a given Boolean formula has exactly km satisfying assignments, for some integer k. We show for all primes p except for possibly one of them, and for all $c < 2\cos(\pi/7) \approx 1.801$, there is a $d > 0$ such that MOD_p-SAT is not solvable in n^c time and n^d space by general algorithms. That is, there is at most one prime p that does not satisfy the tradeoff.
We prove that the same limitation holds for SAT and MOD_6-SAT, as well as MOD_m-SAT for any composite m that is not a prime power. Our main tool is a general method for rapidly simulating deterministic computations with restricted space, by counting the number of solutions to NP predicates modulo integers. The simulation converts an ordinary algorithm into a "canonical" one that consumes roughly the same amount of time and space, yet canonical algorithms have nice properties suitable for counting.

Keywords. time-space tradeoffs, lower bounds, modular counting, satisfiability, reversible computation, diagonalization

Subject classification. 68Q15, 68Q17

1. Introduction

The class $\mathsf{MOD}_m\mathsf{P}$ is the collection of languages L for which there is a nondeterministic polynomial time algorithm that, on input x, has (0 mod m) accepting paths if and only if $x \in L$. The MOD classes naturally formalize the complexity of counting solutions to problems, modulo fixed integers. Recent work by Valiant and others [Val04, Val06, CC06, CL07a, CL07b] has brought the prob-

lems of counting solutions modulo primes (which we call "MOD_p problems") back to the forefront of current research. Surprising algorithms have been found for particular interesting MOD_p problems. A prominent example is the problem of counting satisfying assignments to a planar read-twice monotone 3-CNF formula. In spite of the numerous adjectives restricting the problem, this counting problem is #P-complete [XZZ07]; however, counting the number of satisfying assignments to such a formula *modulo 7* turns out to be in P [Val06, CL07a]. Such results challenge our basic intuitions about the complexity of counting.

In this work we consider the diametric problem of proving concrete limitations on how efficiently MOD_p problems can be solved. Substantial strides have been made recently in discovering deterministic time-space lower bounds for nondeterministic time and the polynomial hierarchy. This work goes back to Kannan [Kan84], who showed (among other results) that $\mathsf{NTIME}[n] \not\subseteq \mathsf{DTISP}[n, o(n)]$. (The result depends on the fact that the space bound in the deterministic computation is small.) In the late 90's, Fortnow [For00] initiated a research program that established time-space lower bounds for the satisfiability problem, showing that $\textsc{Sat} \notin \mathsf{DTIME}[n^{1+o(1)}] \cap \mathsf{NL}$. Fortnow-Lipton-Viglas-Van Melkebeek [LV99, FvM00, FLVV05] improved the tools and arguments, demonstrating that $\textsc{Sat} \notin \mathsf{DTISP}[n^{\phi-\varepsilon}, n^{o(1)}]$ for all $\varepsilon > 0$, where ϕ is the golden ratio. Prior work by the author [Wil06] improved their time lower bound to greater than $n^{\sqrt{3}}$, and Diehl and Van Melkebeek [DvM06] gained an improvement to $n^{1.759}$ by refining the argument. Time-space lower bounds for quantified Boolean formulas with a finite number of quantifier blocks have also been discovered, with larger exponents [FLVV05, Wil06, Wil07a]. All the above results use a form of argument known as *indirect diagonalization* [vM04], which proceeds by showing that the negation of the lower bound one wants to prove implies a contradiction with a known time hierarchy theorem.

Our goal here is to develop methods for extending the results of the previous paragraph to MOD_p problems, in a general way that should also be useful for proving time-space lower bounds on other types of problems in the future. At first, it appears that we might be able to extend such results without much trouble, given that NP can be reduced to $\mathsf{MOD}_p\mathsf{P}$ (for all primes p) by randomized reductions, via the well-known Valiant-Vazirani lemma [VV86]. A quasilinear time version of Valiant and Vazirani's reduction has been found by Naik, Regan, and Sivakumar [NRS95]. Moreover, Toda and Ogihara [TO92] showed that the entire polynomial hierarchy reduces to $\mathsf{MOD}_p\mathsf{P}$ using two-sided randomized reductions (but for $\Sigma_k\mathsf{P}$ where $k \geq 2$, the best reduction we know of from $\Sigma_k\mathsf{TIME}[n]$ to $\mathsf{MOD}_p\mathsf{P}$ takes $\Theta(n^{k+1})$ time, cf. Gupta [Gup98]). However, despite their time-efficiency, the inherent randomness of these reduc-

tions is a major difficulty in applying them to obtain time-space lower bounds, since we do not know how to remove the use of randomness even if we assume that $\mathsf{MOD}_k\mathsf{P}$ problems have efficient algorithms. (We note in passing that, assuming certain unproven circuit lower bounds, deterministic versions of Valiant-Vazirani do exist, cf. [KvM02].)

1.1. Outline of Our Results. We show how to extend the known time-space tradeoffs for SAT to MOD_p-SAT, for *almost* all primes p. In particular, for all distinct primes p and q, we prove that at least one of MOD_p-SAT and MOD_q-SAT exhibits a time-space lower bound, identical to the best known for SAT. Our primary technical contribution, proved in Sections 4 and 5, is a "speedup" theorem showing that deterministic time and space bounded machines can be simulated ultra-efficiently by counting modulo two relatively prime numbers.

Let p and q be integers greater than 1. Informally speaking, we say that a $\mathsf{MOD}_q\mathsf{MOD}_p$ *machine* is a generalization of a nondeterministic machine that begins in a "MOD_q mode", then possibly switches to a "MOD_p mode" at some later point. The machine may also switch to a "deterministic mode", in which case the remaining computation is deterministic. These modes work similarly to the existential and universal modes of an alternating machine: when the machine starts from a MOD_m mode, acceptance occurs if and only if the number of accepting computation paths from the current configuration is *divisible by* m. Let $\mathsf{MOD}_q\mathsf{MOD}_p\mathsf{TIME}[T(n)]$ be the class of sets accepted by $\mathsf{MOD}_q\mathsf{MOD}_p$ machines running in $T(n)$ time. Then our speedup theorem can be stated as follows:

THEOREM 1.1 (Speedup by Modular Counting). *Let M be a deterministic machine running in time T and space S, let $B(n) \leq T(n)$, let $\varepsilon > 0$ be sufficiently small, and let $p, q \geq 2$ be relatively prime. Then there is a $\mathsf{MOD}_q\mathsf{MOD}_p$ machine N such that $L(M) = L(N)$, and N runs for $O(B(n)S(n)\log T(n))$ time in its MOD_q mode, runs for $O(\log(B(n)S(n)\log T(n)))$ time in its MOD_p mode, then runs in deterministic $O(T^{1+\varepsilon}(n)/B(n))$ time and $O(S(n)\log T(n))$ space. Moreover, the number of bits written during previous modes that are read by the final deterministic mode of the computation is only $O(n + S(n)\log T(n))$.*

In our class notation (cf. Section 2.1), we write the above as:

$$\begin{aligned}&\mathsf{DTISP}[T(n), S(n)]\\ &\quad\subseteq\ (\mathsf{MOD}_q\ B(n)S(n)\log T(n))\\ &\qquad(\mathsf{MOD}_p\ \log(B(n)S(n)\log T(n)))\mathsf{DTISP}[\frac{T^{1+\varepsilon}(n)}{B(n)}, S(n)\log T(n)].\end{aligned}$$

By choosing B such that $BS \log T = T^{1+\varepsilon}/B$, we conclude that the class $\mathsf{DTISP}[T(n), S(n)]$ is contained in $\mathsf{MOD}_q\mathsf{MOD}_p\mathsf{TIME}[(T(n)S(n))^{1/2+\varepsilon}]$, for all sufficiently small $\varepsilon > 0$.

Using Theorem 1.1 and some elementary number theory, much of the research on time-space lower bounds for SAT on general models of computation [FLVV05] can be directly transferred over to MOD_m-SAT for various integers m; this "transfer principle" is proved in Section 6.1. Independently of the MOD_m-SAT transfer arguments, we also add a new indirect diagonalization argument to this line of work in Section 6.2, culminating in a new collection of time-space lower bounds. (Note in this paper, we use the notation $g(n) \in \Omega(f(n))$ to mean that $g(n) \geq d \cdot f(n)$ holds *infinitely often*, for some constant $d > 0$.)

THEOREM 1.2. *The following problems require $\Omega(n^c)$ time on random-access machines using $n^{o(1)}$ space, for all c satisfying $c^3 - c^2 - 2c + 1 < 0$,* i.e. *$c < 2\cos(\pi/7)$:*

- SAT, *the satisfiability problem for Boolean CNF formulas.*
- MOD_p-SAT, *the problem of counting the number of satisfying assignments to a Boolean formula modulo p, for all primes p* except for possibly one of them,
- MOD_m-SAT, *for all integers m that are not prime powers.*

The time lower bound implied by Theorem 1.2 is $\Omega(n^{1.801\ldots})$, which is the largest known to date, even for satisfiability. We remark that the lower bound holds not only for $n^{o(1)}$ space but also for n^d space, for sufficiently small $d > 0$ dependent on c. An identical theorem can be stated for any NP problem with sufficiently efficient parsimonious reductions from SAT. More details on this point are provided in the next section.

Unfortunately, Theorem 1.2 does not tell us which one of the primes might be the exception in the MOD_p-SAT lower bound. In order for our argument to become constructive, it seems that we would need to prove a hypothesized time hierarchy theorem.

HYPOTHESIS 1.3. *For all primes p, there exists a prime $q \neq p$ and time constructible $T(n) \geq n^2$ such that $\mathsf{MOD}_p\mathsf{TIME}[T] \not\subseteq \mathsf{MOD}_q\mathsf{TIME}[T^{1-\varepsilon}]$, for all $\varepsilon > 0$.*

(For definitions, cf. Section 2.) The hypothesis looks very reasonable in light of current knowledge, such as the circuit lower bounds for computing MOD_p

with OR, AND, NOT, and MOD_q gates [Smo87]. It seems extremely counterintuitive that counting solutions modulo one prime would somehow always be faster, if one could count modulo a different prime. We show in Section 7 that proving the hypothesis for some prime p would imply a lower bound for MOD_p-SAT.

THEOREM 1.4. *If Hypothesis 1.3 holds for a given prime p, then MOD_p-SAT requires $\Omega(n^c)$ time on $n^{o(1)}$ space machines, for $c < \phi \approx 1.618$, the golden ratio.*

Finally, in Section 8 we use the transfer principle to prove an unconditional separation of complexity classes. Informally speaking, we prove that the class SC is not equal to any unbounded level of the MOD_m hierarchy for polynomial time, for every composite m that is not a prime power.

1.2. Other Related Work.

Modulo Counting Classes. The class $\mathsf{MOD}_2\mathsf{P}$ (also written as $\oplus\mathsf{P}$) was introduced by Papadimitriou and Zachos [PZ83], and the class $\mathsf{MOD}_k\mathsf{P}$ for arbitrary k was introduced by Cai and Hemachandra [CH90]. Beigel and Gill [BG92] showed several closure properties hold for these classes—for example, for all primes p, the class $\mathsf{MOD}_p\mathsf{P}$ is closed under union, intersection, and complement. Thus it is strongly believed that $\mathsf{MOD}_p\mathsf{P} \neq \mathsf{NP}$ for all primes p.

In contrast to the Valiant-Vazirani lemma and Toda's theorem, which show the power of $\mathsf{MOD}_k\mathsf{P}$, Beigel, Buhrman, and Fortnow [BBF98] demonstrated a very interesting oracle collapse/separation for MOD classes. A consequence of their oracle construction is that for all distinct primes p and q, there is an oracle A such that $\mathsf{P}^A = \mathsf{MOD}_p\mathsf{P}^A$, yet $\mathsf{MOD}_q\mathsf{P}^A = \mathsf{NP}^A = \mathsf{EXP}^A$. That is, there are relativized worlds where for any primes p and q, counting NP solutions modulo p is easy, yet counting NP solutions modulo q is as hard as exponential time.

Circuit Lower Bounds with Composite Gates. There has been substantial work on trying to prove lower bounds for circuits with MOD_6 gates, and in general, circuits with MOD_{pq} gates where p and q are distinct primes [BST90, Gro01, CGPT06]. Circuits with such gates appear to be possibly far more powerful than circuits with merely AND, OR, NOT, and MOD_p gates (for some fixed prime p). Our work shows the power of MOD_{pq}-SAT, as the known proofs of lower bounds for SAT can be extended to it.

Lower Bounds for the Counting Hierarchy. Several interesting lower bounds on counting problems have been discovered in the past. Allender and Gore [AG94] proved that uniform ACC^0 is properly contained in PP. Caussinus *et al.* [CMTV98] showed that uniform ACC^0 is properly contained in MODPH (a counting version of the polynomial hierarchy). Allender [All99] showed that the Permanent is not in TC^0, thus $\mathsf{TC}^0 \neq \mathsf{PP}$. Allender *et al.* [AKRRV01] proved time-space tradeoffs for MAJ-MAJ-SAT (MAJ-MAJ-SAT is a complete problem in the second level of the counting hierarchy, a generalization of MAJ-SAT), showing that it is not contained in unbounded-error probabilistic time $n^{1+o(1)}$ and n^{δ} space for all $\delta < 1$.

2. Preliminaries

We assume familiarity with the usual complexity theoretic notions of time, space, and alternation. All functions used to bound runtime and space are assumed to be time constructible within the appropriate space bounds.

2.1. Machine Models. Formally, our underlying machine model shall be the random access Turing machine. The particular details of the machine's inner workings will not affect our results (many other models would work equally well), but for concreteness we include a brief description. We use the *random access Turing machine* model, which has (along with the usual finite control) read-only random access to an input tape, and read-write random access to a finite number of worktapes. Each tape is equipped with an *index tape*, which holds a binary string of length that is logarithmic in the length of the respective tape. We define that the head of a tape is always situated at the location denoted on its corresponding index tape.

Define $\mathsf{DTISP}[T(n), S(n)]$ to be the class of sets accepted by a machine that runs in $T(n)$ time and $S(n)$ space, simultaneously. We shall extensively study the DTISP class in the subpolynomial space ($S(n) = n^{o(1)}$) setting. For convenience, we use the notation

$$\mathsf{DTS}[T(n)] := \mathsf{DTISP}[T(n)^{1+o(1)}, n^{o(1)}]$$

to avoid negligible $o(1)$ additive factors in the exponent. In general, we also follow this convention when writing classes like $\mathsf{NTIME}[n^a]$ or $\Sigma_k\mathsf{TIME}[n^b]$; for the purposes of polynomial-strength lower bounds, these classes are "equivalent" to $\mathsf{NTIME}[n^{a+o(1)}]$ and $\Sigma_k\mathsf{TIME}[n^{b+o(1)}]$, and so we omit these extra $o(1)$ factors except when their inclusion might aid understanding.

Mod Machines. We define a MOD_m *machine* to be a nondeterministic machine with a modified acceptance condition: it accepts an input if and only if the number of its accepting computation paths is divisible by m. It is clear that for any MOD_m machine M running in $O(n^k)$ time, there is a constant c and linear time deterministic machine A such that M accepts a string x iff there are ℓm strings y of length $c|x|^k$ such that $A(x, y)$ accepts, for some integer $\ell > 0$. We define $\mathsf{MOD}_m\mathsf{TIME}[T(n)]$ to be the class of languages accepted by a MOD_m machine in $T(n)$ time.

Building on MOD_m machines, we use an extension of alternating machines introduced by Allender and Gore [AG94], which has not only existential and universal modes but also MOD_m *modes*, for various moduli m. For obvious reasons, we call these *modular-alternating machines*. A state in a MOD_m *mode* acts as one expects it would, in the following sense. Let us assume that the configuration graph for a given alternating machine is a tree. Let σ be a configuration in MOD_m mode where the previous configuration was in a different type of mode (so, an alternation has just occurred). Let T be the maximal subtree of configurations, rooted at σ, where no alternation takes place. Then σ is accepting iff the number of leaves of T that are accepting is divisible by m.

We point out that this notion is somewhat subtle: note that it makes sense for a machine to be in a MOD_6 mode, then alternate to *another*, *different* MOD_6 mode, and that such a machine may be more powerful than a MOD_6 machine. This is because we do not know how to feasibly "merge" consecutive MOD_6 modes into one, as is possible with existential, universal, and MOD_p modes where p is prime. (In fact, a formalization of this question is studied in Theorem 8.1.)

Let us suppose that an alternating machine alternates to specific modes only at specific timesteps, irrespective of the input or current configuration. We say that the *signature* of such a machine is a chronological description of the modes taken during any path of the computation. Typically, the pseudocode for an alternating machine contains commands of the form "*Existentially guess x of length b*" and "*Universally guess x of length b*", where x is some string and b is a positive integer. (These commands mean that the machine is switched to that particular mode and x is chosen over all possible strings of length b.) In a similar manner, our pseudocode for a machine with MOD_k in its signature uses the command "*Modulo k guess x of length b*" analogously.

Notation for Alternating Classes. To precisely describe classes that capture alternating and modular-alternating computations, we use an unorthodox

but natural notation that we have found convenient for our arguments. Define $(\exists\ f(n))\mathcal{C}$ to be the class of languages recognized by a nondeterministic machine N that, on input x, writes a $f(n)^{1+o(1)}$ bit string y nondeterministically and feeds the input $\langle x, y\rangle$ to a machine from class $\mathcal{C}$. The classes $(\forall\ f(n))\mathcal{C}$ and $(\mathsf{MOD}_m\ f(n))\mathcal{C}$ are defined analogously (with co-nondeterministic machines and MOD_m machines, respectively). Note that when (for example) $\mathcal{C} = \mathsf{DTIME}[n^k]$, the n refers to the *original* input length, which could potentially be different from the length of the input $\langle x, y\rangle$ fed to $\mathcal{C}$. So for example, a language is in $(\mathsf{MOD}_2\ n^2)\mathsf{DTIME}[n^4]$ if there is an algorithm A that, given inputs $x \in L$ of length n, there are an odd number of y's of length $n^{2+o(1)}$ such that $A(x, y)$ accepts in $O(n^4)$ time.

When a machine recognizing a language in class $(\exists\ f(n))\mathcal{C}$ is guessing its $f(n)^{1+o(1)}$ bits, we say that the machine is *in an existential quantifier*. Similarly, one can define being *in a universal quantifier* for $(\forall\ f(n))\mathcal{C}$, and being *in a* MOD_m *quantifier* for $(\mathsf{MOD}_m\ f(n))\mathcal{C}$.

2.2. The Completeness of MOD_p-SAT. Our lower bound for MOD_p-SAT first proves that $\mathsf{MOD}_p\mathsf{TIME}[n] \not\subseteq \mathsf{DTS}[n^c]$ for a constant $c > 1$, then argues that this implies MOD_p-SAT is not in $\mathsf{DTS}[n^{c-o(1)}]$. This sort of strategy (exploiting the completeness of a problem) has been invoked in other lower bound arguments as well [FLVV05, AKRRV01, vMR05]. In particular, we apply the following result:

THEOREM 2.1 ([FLVV05, Tou01]). $\text{SAT} \in \mathsf{DTS}[n^c] \implies \mathsf{NTIME}[n] \subseteq \mathsf{DTS}[n^c]$.

THEOREM 2.2. *Let $m \geq 2$ be an integer. If MOD_m-SAT $\in \mathsf{DTS}[n^c]$, then $\mathsf{MOD}_m\mathsf{TIME}[n] \subseteq \mathsf{DTS}[n^c]$.*

PROOF. (Sketch) The proof of Theorem 2.1 gives a reduction from an arbitrary $L \in \mathsf{NTIME}[n]$ to SAT that takes a string x and converts it to an equivalent formula that is of length at most $|x| \cdot \text{poly}(\log |x|)$, and each bit of the formula can be computed individually in $\text{poly}(\log |x|)$ time. From this, the implication

$$\text{SAT is in } n^c \text{ time and } n^{o(1)} \text{ space } \implies \mathsf{NTIME}[n] \subseteq \mathsf{DTS}[n^{c+o(1)}]$$

follows by performing the appropriate reduction and executing the presumed SAT algorithm. We point out that the reduction above is actually *parsimonious*: that is, the number of witnesses for an $x \in L$ equals the number of satisfying

assignments to the resulting formula. From this property, the theorem follows immediately.

For completeness, the remainder of our argument describes why the reduction is parsimonious. From a deterministic linear time algorithm $M(\cdot,\cdot)$, there is a reduction and a constant c with the following properties on a given input x: the reduction produces a CNF formula $\phi_{M,x}(y,z)$, and there is a bit string y (representing a witness for M on x) of length $c|x|$ making $M(x,y)$ accept iff the same y (construed as a Boolean variable assignment) along with some z of length $c|x| \cdot \mathrm{poly}(\log|x|)$ satisfies $\phi_{M,x}(y,z)$. Each such valid z encodes the computation history of the deterministic algorithm M running on (x,y), and is therefore *unique* with respect to a given (x,y) pair. In more detail, z encodes $k+1$ sequences of tuples, where k is the number of tapes of M. Each tuple encodes a snapshot of the computation at some time step: the current state, the symbols read at the beginning of the step, the position of a particular tape head, and the symbols that are written. The first tuple sequence is in *temporal order*, having the form

$$\{\langle i, \text{state, symbol read, head position, symbol written}\rangle\},$$

for time steps $i = 1,\ldots,c|x|$ and a head position on one of the k tapes. (We have k tuples for each time step i, one tuple for each tape.) For $\ell = 2,\ldots,k+1$, the ℓth tuple sequence represents the *spatial order* of the $(\ell-1)$st tape, having the form

$$\{\langle i, \text{state, symbol read}, j, \text{symbol written}\rangle\},$$

for tape positions $j = 1,\ldots,c|x|$ on the $(\ell-1)$st worktape, where the i are increasing when j is fixed. That is, there can be multiple tuples indexed by j, ordered by increasing i. Each tuple takes $O(\log|x|)$ bits to encode, as each index tape holds an $O(\log|x|)$ bit string. The precise function of $\phi_{M,x}(y,z)$ is to check that:

(a) the k tuples for the first step are all in an initial state,

(b) the k tuples for the ith step lead to the k tuples for the $(i+1)$st step via some valid transition (for all i),

(c) the k tuples for the last step are all in a common accepting state, and

(d) all symbols claimed to be read are indeed valid (the first time the ith symbol is read, it is the ith bit of the input if $i \le |x|$, or a blank if $i > |x|$; when a symbol is written to a cell, it is also read later from the same cell).

Conditions (a) and (c) can be easily checked with a short CNF formula over a small set of variables in the temporally ordered sequence of tuples. Condition (b) can also be checked with a short CNF formula over the variables encoding the temporally ordered tuple sequence. To determine condition (d), one formula checks that the temporally ordered sequence is a permutation of the k spatially ordered sequences (this is the most technical step, which can be achieved by using efficient sorting networks [vM07]), and another formula uses the variables of the spatially ordered sequence to verify that (1) when a tape cell is first read, it contains the appropriate value, and (2) every symbol read in a tape cell was the last symbol previously written in that cell. The upshot is that values of all variables in these formulas are completely determined by the tuple sequences, as these formulas encode deterministic computations (in particular, they encode the evaluations of certain circuits on fully specified inputs).

Therefore for any particular guess y for $M(x, y)$, there is precisely one valid setting for the tuple sequences, so there is a one-to-one correspondence between y's that make $M(x, y)$ accept, and (y, z) pairs that satisfy the formula $\phi_{M,x}(y, z)$. Hence the number of valid y such that $M(x, y)$ accepts is divisible by m iff the number of (y, z) satisfying $\phi_{M,x}$ is divisible by m, so the above type of reduction also serves as a reduction from an arbitrary $L \in \mathsf{MOD}_m\mathsf{TIME}[n]$ to MOD_m-SAT. □

3. Intuition Behind This Work

The manner in which our results are proved is perhaps more interesting than the results themselves. Over the next two sections, we shall prove the Speedup By Modular Counting Theorem (Theorem 1.1), which gives a new method for simulating deterministic computations with restricted time and space, via counting. We begin in Section 4 by showing that every machine M has an equivalent *canonical* version $\hat{M}$. The canonical version uses roughly the same time and space as M; however, the task of counting particular properties of $\hat{M}$'s configurations is much easier. The canonical machine simulation follows from an efficient reversible simulation of irreversible computation, first given by Bennett [Ben89]. Our efficient procedure for simulating $\hat{M}$ on an input x counts the number of certain objects (which we call "complete configuration sequences with a number of mistakes divisible by p" for an integer $p \geq 2$) efficiently with one call to a $\mathsf{MOD}_p\mathsf{TIME}[n]$ oracle. In Section 5, we prove that the number of such objects counted in the accepting case is congruent to 1 *modulo* q, and the number counting in the rejecting case is 0 *modulo* q, for $q \geq 2$ relatively prime to p. Thus one can tell the difference between the accepting and rejecting cases

by counting modulo q.

We apply Theorem 1.1 in the remaining sections, showing how to transfer diagonalization arguments establishing time-space tradeoffs for nondeterminism directly to time-space tradeoffs for counting solutions modulo integers, by substituting nondeterminism with "counting mod p", and conondeterminism with "counting mod q". For example, Lipton and Viglas [LV99] proved an $\Omega(n^{\sqrt{2}-\varepsilon})$ time lower bound for solving SAT on $n^{o(1)}$-space machines. One way to phrase their argument uses the fact (proved in Theorem 5.1) that $\mathsf{DTISP}[t,s] \subseteq \Sigma_2\mathsf{TIME}[(ts)^{1/2}]$ in the following way: assuming $\mathsf{NTIME}[n] \subseteq \mathsf{DTS}[n^c]$, one can prove

$$\mathsf{DTS}[n^2] \subseteq \Sigma_2\mathsf{TIME}[n^{1+o(1)}] \subseteq \mathsf{NTIME}[n^{c+o(1)}] \subseteq \mathsf{DTS}[n^{c^2}].$$

(The first inclusion follows from the fact; the last two inclusions follow by assumption and standard padding arguments.) For $c^2 < 2$, the above inclusion implies a contradiction by a non-trivial diagonalization argument (cf. Section 6.2).

We call the above kind of lower bound proof an *alternation-trading proof*, and show that any such proof can be modified so that it works for modular counting rather than nondeterminism. To illustrate, an analogous lower bound for $\mathsf{MOD}_6\mathsf{TIME}[n]$ can be obtained by assuming $\mathsf{MOD}_6\mathsf{TIME}[n] \subseteq \mathsf{DTS}[n^c]$, and proving for all $\varepsilon > 0$ that

$$\begin{aligned}\mathsf{DTS}[n^2] &\subseteq (\mathsf{MOD}_3\ n^{1+\varepsilon})(\mathsf{MOD}_2\ n^{1+\varepsilon})\mathsf{DTS}[n^{1+\varepsilon}] \\ &\subseteq (\mathsf{MOD}_3\ n^{1+\varepsilon})\mathsf{DTS}[n^{c(1+\varepsilon)}] \\ &\subseteq \mathsf{DTS}[n^{c^2(1+\varepsilon)}].\end{aligned}$$

The first inclusion follows from the Speedup by Modular Counting Theorem (Theorem 1.1), and the second and third inclusions follow from the observations that $\mathsf{MOD}_2\mathsf{TIME}[n] \subseteq \mathsf{MOD}_6\mathsf{TIME}[n] \subseteq \mathsf{DTS}[n^c]$, and $\mathsf{MOD}_3\mathsf{TIME}[n] \subseteq \mathsf{MOD}_6\mathsf{TIME}[n] \subseteq \mathsf{DTS}[n^c]$. Again, for $c^2 < 2$, we can obtain a contradiction when ε is sufficiently small. The above kind of transfer can be carried out on other time-space tradeoff arguments similarly.

4. Canonical Machines

We begin by recalling the definition of a configuration graph. Let M be a deterministic machine running in time $T(n)$ and space $S(n) \geq \log n$, and let x be an input string.

DEFINITION 4.1. *The* configuration graph $G_{M,x}$ *of* $M(x)$ *has* $2^{cS(n)}$ *nodes for a sufficiently large integer* $c > 1$*, where every node of the graph is uniquely labeled by a* $cS(n)$*-bit string. There is an arc from the node labeled* v *to the node labeled* w *if and only if, when* v *and* w *are construed as configurations of* M*, the configuration* w *is obtained by executing* M *on* x *in configuration* v *for one step.*

Since M is deterministic, observe that the outdegree of any node in $G_{M,x}$ is at most one. Notice that our definition of configuration graph includes *all* possible strings of length $cS(n)$ as nodes, so many of the nodes do not correspond to legitimate machine configurations. Those nodes will have indegree and outdegree zero.

For our simulations of DTS, we require the machine being simulated to have a configuration graph of a very regular type, which we call *canonical*. It turns out that a deterministic small-space machine can be converted into a canonical one, without a significant increase in time and space usage.

DEFINITION 4.2. *A machine* M *is* canonical *iff for all inputs* x*, every node in the configuration graph* $G_{M,x}$ *has outdegree and indegree exactly one. That is, the graph is a union of disjoint cycles along with isolated nodes having self-loops.*

By definition, a canonical machine does not halt, since no configuration has outdegree zero. Thus we need to modify the acceptance condition for a canonical machine.

DEFINITION 4.3. *Let* M *be canonical.* M accepts input x *if and only if* $M(x)$ *(started in its initial configuration) reaches a configuration with an "accept" state before it reaches its initial configuration again, or before it reaches a rejecting configuration.*

A canonical machine is obviously deterministic, but it also enjoys the following useful property.

PROPOSITION 4.4. *Let* M *be canonical, let* c *be a node of* $G_{M,x}$*, and let* t *be a non-negative integer. Then there are unique configurations* c'_t *and* c''_t *such that, when* $M(x)$ *is executed from* c'_t *for* t *steps,* $M(x)$ *ends in configuration* c*, and when* $M(x)$ *is executed from* c *for* t *steps,* $M(x)$ *ends in configuration* c''_t*.*

Proposition 4.4 *does not hold* for deterministic machines in general, as there can be numerous configurations that lead to a common configuration by executing each one for t steps, and the number of such configurations can, in principle,

depend upon the given input. The "unique predecessor" and "unique successor" properties of canonical machines shall prove to be indispensable in our later simulations of space-bounded computation.

4.1. Making Machines Canonical. In order to show that every deterministic machine can be turned into an equivalent canonical one, we apply a theorem of Bennett that shows how to convert any deterministic machine into one that is *reversible*, with negligible penalty to its time and space complexity. Recall the definition of *reversible machine*: a deterministic machine is reversible if and only if its configuration graph on all inputs is such that all nodes have indegree at most one and outdegree at most one.

THEOREM 4.5 (Bennett [Ben89]). *Let M be a deterministic machine running in time $T(n)$ and space $S(n)$, where $T^{\varepsilon}(n) \geq S(n)$ for all $\varepsilon > 0$. Then for every integer $k \geq 1$, there is a reversible machine M' that runs in $O(T^{1+1/k}(n))$ time and $O(S(n) \log_2 T(n))$ space, such that $L(M) = L(M')$.*

Bennett originally stated his result in terms of Turing machines, but it works equally well for random access machines. We outline the proof of Theorem 4.5 in the following paragraphs.

Suppose M is a deterministic machine to be simulated on input x, and it runs in time T and space S. We assume without loss of generality that M has a unique accepting configuration. The most naïve way to simulate M reversibly would be to store a history of M's configurations on blank tape as the simulation progresses step-by-step. Doing this, there is always a *unique* step in reverse: take the previous configuration written to the configuration list, update M to be back in that configuration, and erase that configuration from the history. However, this simulation would require $\Omega(TS)$ space, so it is unsuitable for us.

The high-level idea behind Bennett's simulation M' is to maintain only $O(\log_2 T)$ past configurations of $M(x)$ in storage, adding and erasing configurations from history in a more intelligent way. Informally, the simulation works as follows: after it reaches a new configuration it hasn't reached before, it begins simulating *backwards*, reversibly erasing previous stored configurations in the process. Finally it continues simulating (forwards) from the new configuration. By storing enough intermediate configurations, the runtime of Bennett's simulation is only $O(T^{1+1/k})$.

In more detail, suppose we want to simulate M for ℓ^m steps for a constant $\ell \geq 2$, starting from a given configuration C. We assume without loss of

generality that the tape alphabet of M' is binary, and that zeroes are analogous to blanks. The procedure M' can be described recursively as follows:

- If $m = 0$ then
  - Simulate M from C for one step, and XOR the configuration D obtained in a designated block of S cells on a tape T_0.
- If $m \geq 1$, then let $C_1 = C$ in the following.
  - For $i = 1, \ldots, \ell$, recursively simulate M from C_i for ℓ^{m-1} steps. Let C_{i+1} be the configuration obtained, and XOR it with the block of S cells immediately to the right of C_i on a tape T_m. (On blank tape, this for-loop has the effect of producing a list of configurations $C_1 C_2 \cdots C_{\ell+1}$ on tape T_m.)
  - For $i = \ell - 1, \ldots, 1$, recursively simulate M from C_i for ℓ^{m-1} steps. Let C_{i+1} be the result, and XOR it with the block of S cells immediately to the right of C_i on T_m. Then, using only XOR operations, move the configuration $C_{\ell+1}$ immediately to the right of C_i on T_m. (This for-loop has the effect of producing just the list $C_1 C_{\ell+1}$ on a tape T_m.)

We have been deliberately vague in writing precise XOR instructions in the above, to keep the description simple. The primary idea is that when the simulation is run once, the XOR operation occurs on blank cells (zeroes); when the simulation is run the second time, the configuration C_i is XORed where it was written the first time, "erasing" that block of cells. At the end of the ℓ^m-step simulation, only the final configuration $C_{\ell+1}$ is stored: the second for-loop simulates from $C_{\ell-1}$, then erases C_ℓ (by XORing), moves $C_{\ell+1}$ into the block where C_ℓ used to be, then simulates from $C_{\ell-2}$, erases $C_{\ell-1}$, moves $C_{\ell+1}$ to where $C_{\ell-1}$ used to be, *etc.*, until only $C_1 C_{\ell+1}$ is on the tape. It is important to observe that if this simulation were run in *reverse* where the tapes consist only of C_1 and blanks, then the "move" instruction of the second for-loop would have no effect. (Technically, the "move" instruction of the second for-loop should also appear analogously in the first for-loop, but we have omitted it for simplicity.)

It is clear that M' faithfully simulates M, since it simulates M step-for-step. Bennett proves that M' is also reversible: at every stage of the computation, the previous step and next step are both uniquely determined. This follows from the reversible properties of the XOR operation, and by the design of the

recursive procedure itself. In more detail, the *reverse* of the above procedure is essentially *itself*: if we perform all the recursive calls in reverse order (starting with the last recursive call of the above procedure, and ending with the first recursive call), we obtain the same set of recursive calls. While the above procedure uses m tapes, such a machine can be implemented with little overhead on a RAM, for small m.

Let us sketch the space and time analysis of the above simulation, following Levine and Sherman [LS90]. Since there are $2\ell - 1$ recursive calls, the runtime of M' is on ℓ^m steps is given by the recurrence:

$$\begin{aligned} T'(\ell^0) &= O(S), \\ T'(\ell^m) &= (2\ell - 1)T'(\ell^{m-1}) + O(S), \end{aligned}$$

or $T'(\ell^m) \leq O((2\ell - 1)^m S)$. As the runtime of M is $T = \ell^m$, by choosing $\ell = \Theta(2^{k'})$ for a given $k' > k$, one can show the runtime of M' is $O((2\ell-1)^m S) \leq O(T^{1+1/k'} S) \leq O(T^{1+1/k'+\varepsilon})$ for all $\varepsilon > 0$, by our assumption on S. Hence for sufficiently small $\varepsilon > 0$, the runtime is $O(T^{1+1/k})$. It can be verified that the space usage of M' for ℓ^m steps is given by the recurrence:

$$\begin{aligned} S'(\ell^0) &= O(S), \\ S'(\ell^m) &= S'(\ell^{m-1}) + O(\ell S). \end{aligned}$$

Therefore the space usage of M' is $O(\ell S \log_\ell T) \leq O(S \log T)$, for any constant k.

Two properties are immediate from the description of M'.

REMARK 4.6. *The reversible machine M' of Theorem 4.5 has the following properties:*

(i) For all positive integers n, there is a unique configuration A_n that can be computed in $O(S(n))$ time such that, on all x of length n, $M'(x)$ accepts $\iff$ $M'(x)$ is in configuration A_n after a prescribed number of steps from its initial configuration.

(ii) There is a machine M'_R such that on all x, $G_{M'_R,x}$ is equivalent to $G_{M',x}$ except that all arcs point in the opposite directions. That is, M' on x goes from configuration C to C' in one step if and only if M'_R on x goes from configuration C' to C in one step.

Applying the above two remarks, it is not difficult to construct an equivalent canonical machine for any deterministic machine that has the same asymptotic time and space complexity as the reversible simulation of Theorem 4.5.

THEOREM 4.7. *Let M be a deterministic machine running in time $T(n)$ and space $S(n)$. For all $\varepsilon > 0$, there exists a canonical machine $\hat{M}$ that runs in $T^{1+\varepsilon}(n)$ time and $O(S(n) \log_2 T(n))$ space, with $L(\hat{M}) = L(M)$.*

PROOF. The first step is to apply Theorem 4.5 with $k > 1/\varepsilon$, turning M into an equivalent reversible machine M'. Let M'_R be the machine guaranteed by Remark 4.6. Since M' is reversible, M'_R is reversible as well. Thus the graphs $G_{M',x}$ and $G_{M'_R,x}$ already have maximum indegrees and outdegrees of at most one. To obtain an equivalent $\hat{M}$ so that every node of some $G_{\hat{M},x}$ has indegree and outdegree *exactly* one, we make the following change.

$\hat{M}$ has two *modes* for each state of M: forward and backward. $\hat{M}$ starts in forward mode from the initial configuration of M', and simulates M' normally. Two rules are applied:

- If $\hat{M}$ is in forward mode and reaches a configuration where no transition applies, then in one transition, $\hat{M}$ switches to backward and starts executing M'_R.
- If $\hat{M}$ is in backward mode and reaches a configuration where no transition applies, then in one transition, $\hat{M}$ switches to forward and starts executing M'.

We claim that $\hat{M}$ is canonical. Notice that $G_{\hat{M},x}$ has precisely double the nodes of $G_{M',x}$, one for each of the two modes of $\hat{M}$. Thus we can label each node of $G_{\hat{M},x}$ as a pair $\langle C, m \rangle$, where C is a configuration of $G_{M',x}$ and m is a mode. Therefore $\hat{M}(x)$ has precisely twice the number of configurations as $M'(x)$.

Let $\langle C, m \rangle$ be a configuration of $G_{\hat{M},x}$. There are four possible cases.

1. C **has outdegree 0 and indegree 0.** (That is, C is not a legal machine configuration.) Then the edges adjacent to $\langle C, \mathsf{forward} \rangle$ and $\langle C, \mathsf{backward} \rangle$ in $G_{\hat{M},x}$ are $(\langle C, \mathsf{forward} \rangle, \langle C, \mathsf{backward} \rangle)$ and $(\langle C, \mathsf{backward} \rangle, \langle C, \mathsf{forward} \rangle)$.
2. C **has outdegree 1 and indegree 1.** Let (C, C') and (C'', C) be the respective edges. Then the adjacent edges for the node $\langle C, \mathsf{forward} \rangle$ are $(\langle C, \mathsf{forward} \rangle, \langle C', \mathsf{forward} \rangle)$ and $(\langle C'', \mathsf{forward} \rangle, \langle C, \mathsf{forward} \rangle)$. Similarly, the node $\langle C, \mathsf{backward} \rangle$ has the adjacent edges $(\langle C', \mathsf{backward} \rangle, \langle C, \mathsf{backward} \rangle)$, and $(\langle C, \mathsf{backward} \rangle, \langle C'', \mathsf{backward} \rangle)$.
3. C **has outdegree 1 and indegree 0.** Let (C, C') be the respective edge. Then the edges adjacent to the node $\langle C, \mathsf{forward} \rangle$ in $G_{\hat{M},x}$ are

$(\langle C, \mathsf{forward}\rangle, \langle C', \mathsf{forward}\rangle)$ and $(\langle C, \mathsf{backward}\rangle, \langle C, \mathsf{forward}\rangle)$. Similarly, $(\langle C, \mathsf{backward}\rangle, \langle C, \mathsf{forward}\rangle)$ and $(\langle C', \mathsf{backward}\rangle, \langle C, \mathsf{backward}\rangle)$ are the edges adjacent to $\langle C, \mathsf{backward}\rangle$ in $G_{\hat{M},x}$.

4. C **has outdegree 0 and indegree 1.** Analogous to the previous case.

Finally, note $M(x)$ accepts in $T(|x|)$ time if and only if $\hat{M}(x)$ accepts in $T^{1+\varepsilon}(|x|)$ time, and $\hat{M}(x)$ uses only $O(S(|x|)\log T(|x|))$ space. □

5. Simulating Space-Bounded Machines By Counting

In this section, we establish Theorem 1.1, which shows that any language in $\mathsf{DTISP}[T(n), S(n)]$ can be simulated extremely efficiently by a machine of signature $\mathsf{MOD}_q\mathsf{MOD}_p$, for any $p, q \geq 2$ that are relatively prime. To do this, we first review a method for simulating any time $T(n)$, space $S(n)$ machine by a Σ_2 machine that runs in $(T(n) \cdot S(n))^{1/2}$ time [Nep70, Kan84]. That is, one can "speed up" a deterministic machine by introducing two quantifiers in the computation, when the machine's time-space product is small. We sketch the construction here for completeness.

Theorem 5.1 (Follows from Nepomnjascii [Nep70], Kannan [Kan84]).

$$\mathsf{DTISP}[T(n), S(n)] \subseteq \Sigma_2\mathsf{TIME}[(T(n)S(n))^{1/2}].$$

Moreover, for every $B(n) \leq T(n)$,

$$\mathsf{DTISP}[T(n), S(n)] \subseteq (\exists\ B(n)S(n))(\forall\ \log B(n)S(n))\mathsf{DTISP}[T(n)/B(n), S(n)].$$

Proof. Let M be a random access machine running in $T(n)$ time and $S(n)$ space. Its simulation $N(x)$ begins by existentially guessing a sequence of $B(|x|)$ configurations of $M(x)$. $N(x)$ then appends the initial configuration to the beginning of the sequence and the accepting configuration to the end of the sequence. Finally, $N(x)$ universally guesses an integer $i \in \{0, \ldots, B(|x|)\}$ and simulates $M(x)$ starting from the ith configuration in the sequence, accepting if and only if the $(i+1)$st configuration in the sequence is reached in $T(|x|)/B(|x|)$ steps. It is easy to see that for all x, $M(x)$ accepts if and only if $N(x)$ accepts. Observe that for $B(n) = \sqrt{T(n)/S(n)}$, we have $\mathsf{DTISP}[T(n), S(n)] \subseteq \Sigma_2\mathsf{TIME}[(T(n)S(n))^{1/2}]$. Furthermore, the existential guess has length $O(B(n)S(n))$, the universal guess has length $O(\log(B(n)S(n)))$, and the remaining deterministic computation runs in $T(n)/B(n)$ time and $S(n)$ space. □

We shall prove that, with some modifications, a simulation akin to the above can successfully simulate canonical machines when the *existential modes* of N are replaced by MOD_q *modes*, and the *universal modes* are replaced by MOD_p *modes*, where p and q are relatively prime. Theorem 1.1 follows from such a simulation. We begin with some definitions. Let machine M run in time $T(n)$, and let $B \geq 1$ in the following.

DEFINITION 5.2. *A* B-configuration sequence for M on x *is a tuple of the form* $\langle C_0, C_1, \ldots, C_B, C_{B+1} \rangle$, *where* C_i *are configurations of* M *on* x.

Such a sequence is called complete *if* C_0 *is the unique initial configuration of* M *and* C_{B+1} *is the unique accepting configuration.*

Such a sequence is correct *if for all* $i = 0, \ldots, B-1$, *when* $M(x)$ *is run for*

$$\lfloor T(|x|)/(B+1) \rfloor + 1$$

steps from C_i, *the resulting configuration is* C_{i+1}, *and when* $M(x)$ *is run for*

$$\max\{T(|x|) - B\left(\lfloor T(|x|)/(B+1) \rfloor + 1\right), 1\}$$

steps from C_B, *the resulting configuration is* C_{B+1}.

If a B*-configuration sequence is not correct then it is said to have a* mistake *at the pair* (C_i, C_{i+1}) *when* C_i *does not result in* C_{i+1} *in the proper number of steps.*

Notice that

$$\begin{aligned} T(|x|) - B\left(\lfloor T(|x|)/(B+1) \rfloor + 1\right) &\leq T(|x|) - BT(|x|)/(B+1) \\ &= T(|x|)/(B+1) \\ &\leq \lfloor T(|x|)/(B+1) \rfloor + 1, \end{aligned}$$

so the last pair (C_B, C_{B+1}) of a configuration sequence requires no more steps than the other pairs. Also observe the number of B-configuration sequences depends solely on the space complexity of the machine.

CLAIM 5.3. *Let* n *and* B *be positive integers, and let* M *be a machine running in space* $S(n)$. *There is a number* $N(n)$ *such that for all* x *satisfying* $|x| = n$, *the number of* B*-configuration sequences for* $M(x)$ *is precisely* $N(n)$.

PROOF. Without loss of generality, the number of configurations for M on inputs of length n is $2^{cS(n)+d}$ for some integers $c > 1$ and $d > 1$ that are independent of n. Thus the total number of B-configuration sequences is $N(n) = 2^{(B+2)(cS(n)+d)}$. □

Clearly, for any input x, the number of B-configuration sequences that are correct and complete is either zero ($M(x)$ rejects) or one ($M(x)$ accepts). The next lemma shows that, for canonical machines, the number of configuration sequences that are merely correct (but not necessarily complete) depends solely on the input length:

LEMMA 5.4. *Let $\hat{M}$ be canonical and let $C(n)$ be the number of configurations of $\hat{M}$ on inputs of length n. Let x be an input and $n = |x|$. Then for every integer B, the number of correct B-configuration sequences for $\hat{M}(x)$ is exactly $C(n)$. In fact, a correct B-configuration sequence is uniquely determined by specifying an integer $i \in \{0, 1, \ldots, B+1\}$ and a configuration c.*

PROOF. For each configuration, there is exactly one correct B-configuration sequence that starts with that configuration, by Proposition 4.4. Moreover, Proposition 4.4 implies that, by specifying even a single configuration c and its position in a correct B-configuration sequence, the rest of the correct B-configuration sequence is uniquely determined, as all configurations prior to c and all configurations after c are unique. □

We now give a way to differentiate between the accepting and rejecting cases for a canonical machine, by counting the mistakes in configuration sequences. The Counting Lemma shows that on any input x, the number of complete B-configuration sequences with k mistakes at adjacent pairs depends only on the input length n, the number of configurations B in the sequence, and whether or not the input leads to acceptance.

LEMMA 5.5 (Counting Lemma). *Let n and $B \geq 1$ be positive integers, let $k = 0, 1, \ldots, B+1$, and let $\hat{M}$ be canonical. Then there are positive integers $N_A(n, k, B)$ and $N_R(n, k, B)$ such that, for all inputs x of length n:*

(i) If $\hat{M}(x)$ accepts, then the number of complete B-configuration sequences for $\hat{M}(x)$ with exactly k mistakes (at adjacent pairs) is $N_A(n, k, B)$.

(ii) If $\hat{M}(x)$ rejects, then the number of complete B-configuration sequences for $\hat{M}(x)$ with exactly k mistakes (at adjacent pairs) is $N_R(n, k, B)$.

The proof shows that $N_A(n, k, B) = \binom{B+1}{k} N_A(n, k)$ and $N_R(n, k, B) = \binom{B+1}{k} N_R(n, k)$, where $N_A(n, k)$ (respectively, $N_R(n, k)$) is defined to be the number of complete B-configuration sequences with mistakes at some fixed set of k adjacent pairs, on the condition that $M(x)$ accepts (respectively, $M(x)$ rejects) and $n = |x|$. Then $N_A(n, k)$ and $N_R(n, k)$ are shown to be well-defined

(that is, they do *not* depend on the k-set that we choose, and they only depend on the input x up to acceptance and rejection). This implies (i) and (ii).

PROOF. Fix a canonical $\hat{M}$ and input x of length n. We count the number of ways that one can choose a complete B-configuration sequence with precisely k mistakes at adjacent pairs.

Define $N_A(n,k)$ (and $N_R(n,k)$, respectively) to be the number of complete B-configuration sequences with mistakes at exactly a particular set of k adjacent pairs, on the condition that $\hat{M}(x)$ accepts (rejects, respectively). Assuming these quantities are well-defined (that is, they depend *only* on n and k), we claim that for all $k \geq 0$,

$$\begin{array}{c} N_A(n,k,B) = \binom{B+1}{k} N_A(n,k) \\ N_R(n,k,B) = \binom{B+1}{k} N_R(n,k). \end{array} \tag{5.6}$$

The equation (5.6) follows since there are $B+1$ possible adjacent pairs for which a mistake can occur, so the total number of possible choices for picking the k points at which a mistake occurs is $\binom{B+1}{k}$.

We now turn to proving that $N_A(n,k)$ and $N_R(n,k)$ are well-defined. The proof is by induction on k. When $k = 0$, we have:

- $N_A(n,0) = 1$. The computation is deterministic, so there is only one complete B-configuration sequence with no mistakes, on the condition that $\hat{M}(x)$ accepts.
- $N_R(n,0) = 0$. When we assume that $\hat{M}(x)$ rejects, there is no complete B-configuration sequence with no mistakes.

For the inductive step, recall that $C(n)$ is the number of configurations of $\hat{M}$ on inputs of length n. We prove for all $k \geq 1$ that

$$\begin{array}{rcl} N_A(n,k) & = & (C(n)-1)^{k-1} - N_A(n,k-1) \\ N_R(n,k) & = & (C(n)-1)^{k-1} - N_R(n,k-1). \end{array} \tag{5.7}$$

Let $i_1, \ldots, i_k \in \{0, \ldots, B\}$ be the specified points for which our B-configuration sequences will make a mistake, with $i_1 < i_2 < \cdots < i_k$. We want to show that the number of complete B-configuration sequences with mistakes precisely at the pairs (C_{i_j}, C_{i_j+1}), for $j = 1, \ldots, k$, is a quantity independent of the set $\{i_1, \ldots, i_k\}$ and the particular input x.

First we claim that the number of complete B-configuration sequences with mistakes at the pairs (C_{i_j}, C_{i_j+1}) for $j = 1, \ldots, k-1$, and no mistakes at the

pairs indexed by $\{0, \ldots, B\} - \{i_1, \ldots, i_k\}$, is exactly $(C(n)-1)^{k-1}$. (Note we allow for a mistake to possibly happen at the pair (C_{i_k}, C_{i_k+1}).) This is due to Proposition 4.4: for every configuration C_{i_j}, there are $C(n)-1$ configurations that it does not lead to, for any prescribed number of steps.

However, this quantity is overcounting—we need to subtract out those configuration sequences with a mistake at the first $k-1$ pairs, but no mistake anywhere else. But this quantity is $N_A(n, k-1)$ if $\hat{M}(x)$ accepts, and $N_R(n, k-1)$ if $\hat{M}(x)$ rejects, by the induction hypothesis. Therefore the equation (5.7) holds, and the quantities $N_A(n,k)$, $N_R(n,k)$ are well-defined for all n and k. □

The Counting Lemma gives exact values for the number of mistakes in configuration sequences. One might ask if we can simulate a canonical machine by simply counting, over all complete B-configuration sequences, all the mistakes made by adjacent pairs. By choosing $B = (T(n)/S(n))^{1/2}$, such a simulation could be implemented in roughly $(T(n)S(n))^{1/2}$ time as a #P function using an argument like that of Theorem 5.1, yielding a speedup of a time $T(n)$, space $S(n)$ machine by a #P function.

However, the total number of mistakes committed by all adjacent pairs over all complete B-configuration sequences is the *same* in the accepting and rejecting cases, so the above proposal does not work. (To see this, note that the quantity is the sum over all $j = 0, \ldots, B$ of the number of complete B-configuration sequences having a mistake at (C_j, C_{j+1}). For every fixed j, this number of sequences is $C(n)^{B-1}(C(n)-1)$, so it does not depend on acceptance or rejection.) Instead, we count the number of complete B-configuration sequences where the number of mistakes is *divisible by some integer* $p \geq 2$, and compute this number of sequences modulo another integer q that is relatively prime to p. It turns out that this is enough to distinguish between the accepting and rejecting cases. We first need a simple lemma.

LEMMA 5.8. *Let $p, q \geq 2$ be relatively prime, let $\ell > 0$ be an integer, and let $k = 0, \ldots, \lfloor \frac{q^\ell}{p} \rfloor$. If $k = 0$, then $\binom{q^\ell}{kp} \equiv 1 \mod q$; if $k > 0$, then $\binom{q^\ell}{kp} \equiv 0 \mod q$.*

PROOF. When $k = 0$, the lemma is obvious. Let $k > 0$. We have

$$kp \cdot \binom{q^\ell}{kp} = q^\ell \cdot \binom{q^\ell - 1}{kp - 1}.$$

(Both sides count the number of ways that one can choose kp elements from a collection of q^ℓ, and place a mark on one of the kp elements.) Note that $0 < kp \leq \lfloor q^\ell/p \rfloor p < q^\ell$, where the last inequality is strict because p and q are

relatively prime. Therefore, there is a non-trivial factor of q^ℓ that divides $\binom{q^\ell}{kp}$, and thus $\binom{q^\ell}{kp} \equiv 0 \mod q$. □

(Notice that the lemma holds under an even weaker condition, namely that p does not divide q^ℓ for every $\ell > 0$. Since it is unclear what the power of this weaker condition is, to keep it simple we just assume that p and q are relatively prime.) For any positive integers B, p, and q, define

$$\mathsf{MISTAKES}_A(n, B, p, q) := \left(\sum_{k=0}^{\lfloor (B+1)/p \rfloor} N_A(n, kp, B) \right) \mod q$$

and

$$\mathsf{MISTAKES}_R(n, B, p, q) := \left(\sum_{k=0}^{\lfloor (B+1)/p \rfloor} N_R(n, kp, B) \right) \mod q.$$

These quantities denote the number (modulo q) of complete B-configuration sequences where the number of mistakes is divisible by p, in the case where the underlying machine accepts or rejects, respectively. In order for a $\mathsf{MOD}_q\mathsf{MOD}_p$ simulation of this kind to work, we need to know the residue modulo q that we "expect" to see in the accepting case, and we need this residue to be different from the rejecting case. The next lemma satisfies these requirements.

LEMMA 5.9. *Let $B = q^\ell - 1$, for an integer $q \geq 2$ and positive integer ℓ. Then for every $p \geq 2$ relatively prime to q,*

$$\mathsf{MISTAKES}_A(n, B, p, q) \equiv 1 \mod q, \quad \mathsf{MISTAKES}_R(n, B, p, q) \equiv 0 \mod q.$$

PROOF.

$$\begin{array}{rcll} \mathsf{MISTAKES}_A(n, B, p, q) & = & \sum_{k=0}^{\lfloor (B+1)/p \rfloor} N_A(n, kp, B) & \\ & = & \sum_{k=0}^{\lfloor (B+1)/p \rfloor} N_A(n, kp) \binom{B+1}{kp} & \text{(Lemma 5.5)} \\ & = & \sum_{k=0}^{\lfloor q^\ell/p \rfloor} N_A(n, kp) \binom{q^\ell}{kp}. & \end{array}$$

But $\binom{q^\ell}{kp} \equiv 0 \mod q$ for all $k = 1, \ldots, \lfloor q^\ell/p \rfloor$ (Lemma 5.8) and when $k = 0$, $N_A(n, 0)$ and $\binom{q^\ell}{kp}$ are both $1 \mod q$, by Lemmas 5.5 and 5.8, respectively. Therefore $\mathsf{MISTAKES}_A(n, B, p, q) \equiv 1 \mod q$. An analogous argument shows that $\mathsf{MISTAKES}_R(n, B, p, q) \equiv 0 \mod q$ (the only difference being that $N_R(n, 0) = 0$). □

Combining the two previous lemmas, we can immediately conclude that for an input x of length n and appropriate p, q, and B:

- if $\hat{M}(x)$ accepts, then the number (modulo q) of complete B-configuration sequences *with its number of mistakes divisible by* p is 1, and
- if $\hat{M}(x)$ rejects, then the same quantity is 0.

We finally arrive at the proof of the Speedup by Modular Counting Theorem (Theorem 1.1). Recall its statement:

Theorem 1.1 *Let M be a deterministic machine running in time T and space S, let $B(n) \leq T(n)$, let $\varepsilon > 0$ be sufficiently small, and let $p, q \geq 2$ be relatively prime. Then there is a $\mathsf{MOD}_q\mathsf{MOD}_p$ machine N such that $L(M) = L(N)$, and N runs for $O(B(n)S(n)\log T(n))$ time in its MOD_q mode, runs for $O(\log(B(n)S(n)\log T(n)))$ time in its MOD_p mode, then runs in deterministic $O(T^{1+\varepsilon}(n)/B(n))$ time and $O(S(n)\log T(n))$ space. Moreover, the number of bits written during previous modes that are read by the final deterministic mode of the computation is only $O(n + S(n)\log T(n))$.*

Proof of Theorem 1.1. Given M that runs in time $T(n)$ and space $S(n)$, let $\hat{M}$ be the efficient canonical version of M guaranteed by Theorem 4.7 that runs in time $T'(n) = T^{1+\varepsilon}(n)$ and space $S'(n) = S(n)\log T(n)$. Let I_n be the unique initial configuration for $\hat{M}$ on inputs of length n. We may also assume that $\hat{M}$ has a unique accepting configuration: by Remark 4.6, the reversible version M' of M has a unique accepting configuration A_n on inputs of length n. Then define $\hat{A}_n = \langle A_n, \mathsf{forward}\rangle$ to be the unique accepting configuration for $\hat{M}$.

Informally, the idea is to run the Σ_2 simulation of $\hat{M}$ from Theorem 5.1, but we replace the existential mode with a "MOD_q mode" and the universal mode with a "MOD_p mode".

Fix an input x so that $\hat{M}(x)$ runs in $T'(|x|)$ time. Let ℓ be the smallest integer such that $B(|x|) \leq q^\ell$. Pseudocode for the simulation of $\hat{M}$ by a $\mathsf{MOD}_q\mathsf{MOD}_p$ machine follows:

(0) *Set $B' = q^\ell - 1$.*

(1) *Modulo q guess either one of $q-1$ dummy accepting paths, or a list of B' configurations $C_1, \ldots, C_{B'}$ of $\hat{M}(x)$.*

Let $C_0 = I_n$ and $C_{B'+1} = \hat{A}_n$.

(2) *Modulo p guess $i = 0, \ldots, B'$.*

If $i < B'$ then $s := \lfloor T'(|x|)/(B'+1) \rfloor + 1$,
else $s := \max\{1, T'(|x|) - B \cdot (\lfloor T'(|x|)/(B'+1) \rfloor + 1)\}$.

Accept if and only if $\hat{M}(x)$ run from configuration C_i for s steps does not result in C_{i+1}.

First, note that $B' = \Theta(B(n))$. We claim that Steps (1) and (2), taken as a computation with signature $\mathsf{MOD}_q\mathsf{MOD}_p$, accepts if and only if $\hat{M}(x)$ accepts. Step (2) accepts if and only if the number of mistakes in the given sequence $C_0, \ldots, C_{B'+1}$ is divisible by p. Let t be the total number of complete B'-configuration sequences for $\hat{M}(x)$ with a number of mistakes divisible by p. Then Steps (1) and (2) accept if and only if

$$q - 1 + t \equiv 0 \mod q,$$

which is equivalent to $t \equiv 1 \mod q$, *i.e.* $t \equiv \mathsf{MISTAKES}_A(n, B', p, q) \mod q$ (by Lemma 5.9), which is true if and only if $\hat{M}(x)$ accepts.

Finally, observe that the deterministic part of the above computation (the last line of Step (2)) requires only $O(T'(n)/(B'+1)) \leq O(T'(n)/B(n))$ time and $O(S'(n))$ space. The deterministic part only takes x and two configurations as input, so its input has length $O(n + S'(n))$. □

6. Lower Bounds for Counting Solutions Modulo Composites

We are now prepared to prove lower bounds. In this section we define the notion of an *alternation-trading proof*, which encompasses all known proofs to date of time-space lower bounds for satisfiability and quantified Boolean formulas. Then we show that alternation-trading proofs can always be transformed into lower bounds for MOD_m-SAT, for certain choices of m. This is done by using simple properties of modular arithmetic, as well as our Speedup by Modular Counting Theorem. One of our results is that there exists *at most one* prime p for which $\mathsf{MOD}_p\mathsf{TIME}[n] \subseteq \mathsf{DTISP}[n^c, n^{o(1)}]$ can hold, when $c < 1.801$. In fact, something stronger can be shown: we demonstrate a *transfer principle* that shows how to translate certain time-space lower bound proofs for $\mathsf{NTIME}[n]$ into analogous arguments for $\mathsf{MOD}_p\mathsf{TIME}[n]$. Intuitively, the statement of the transfer principle says that if efficient SAT algorithms imply a contradiction via indirect diagonalization, then efficient MOD_p-SAT and MOD_q-SAT algorithms also lead to a contradiction. Finally, we prove a new time lower bound for $\mathsf{NTIME}[n]$ in the subpolynomial ($n^{o(1)}$) space setting.

DEFINITION 6.1. *Let $c > 1$, and let $r(n)$, $s(n)$ be polynomials. An* alternation-trading proof *of*

$$\mathsf{NTIME}[n] \subseteq \mathsf{DTS}[n^c] \Longrightarrow \mathsf{DTS}[r(n)] \subseteq \mathsf{DTS}[s(n)]$$

is a list of complexity classes of the form

$$(Q_1\ n^{a_1})(Q_2\ n^{a_2}) \cdots (Q_k\ n^{a_k})\mathsf{DTS}[n^{a_{k+1}}],$$

where $Q_i \in \{\exists, \forall\}$, k is a non-negative integer, and $a_i > 0$ for all i. (When $k = 0$, the class is deterministic.) The first and last classes on the list are $\mathsf{DTS}[r(n)]$ and $\mathsf{DTS}[s(n)]$, respectively. Each class on the list is obtained from the previous class by applying one of three rules:

(i) **(Quantifier Speedup)** *For all $B \geq 1$ and $T \geq B$,*

$$\begin{array}{l} \mathsf{DTS}[T] \subseteq (\exists\ B \cdot n^{o(1)})(\forall\ \log T)\mathsf{DTS}[T/B] \\ \mathsf{DTS}[T] \subseteq (\forall\ B \cdot n^{o(1)})(\exists\ \log T)\mathsf{DTS}[T/B], \end{array} \quad \textit{(Theorem 5.1)}$$

where the DTS *predicate of the alternating simulation reads only $n^{o(1)}$ bits of the first quantifier. This rule is applied by replacing the* DTS *predicate in a class by a two-quantifier simulation of it.*

(ii) **(Quantifier Removal)** *For all $b \geq a \geq 1$,*

$$\mathsf{NTIME}[n] \subseteq \mathsf{DTS}[n^c] \implies \begin{array}{l} (\exists n^a)(\forall n^b)\mathsf{DTS}[n^b] \subseteq (\exists n^a)\mathsf{DTS}[n^{bc}] \\ (\forall n^a)(\exists n^b)\mathsf{DTS}[n^b] \subseteq (\forall n^a)\mathsf{DTS}[n^{bc}] \end{array}.$$

This rule is applied by replacing the last quantifier and DTS *predicate of a complexity class with one quantifier and the appropriate* DTS *predicate.*

(iii) **(Quantifier Combination)** *For all $a, b \geq 0$ and $d \geq 1$,*

$$\begin{array}{l} (\exists\ n^a)(\exists\ n^b)\mathsf{DTS}[n^d] \subseteq (\exists\ n^a + n^b)\mathsf{DTS}[n^d] \\ (\forall\ n^a)(\forall\ n^b)\mathsf{DTS}[n^d] \subseteq (\forall\ n^a + n^b)\mathsf{DTS}[n^d]. \end{array}$$

This rule is applied by replacing adjacent quantifiers of the same type by a single quantifier.

Note that the Quantifier Removal rule follows by a simple padding argument; informally speaking, we need $b \geq a \geq 1$ because one can "pad up", but not "down".

Every known (model-independent) time-space lower bound for SAT can be written as an alternation-trading proof that shows $\mathsf{NTIME}[n] \subseteq \mathsf{DTS}[n^c] \Longrightarrow \mathsf{DTS}[r(n)] \subseteq \mathsf{DTS}[s(n)]$, for polynomials r and s such that $r(n)/s(n) \geq \Omega(n^\delta)$ for some $\delta > 0$. (An example of such a proof can be found in Section 3.)

6.1. The Transfer Principle. We now state the transfer principle, which effectively shows that the alternating classes in an alternation-trading proof can be replaced with *modular-alternating* classes:

Transfer Principle for Time-Space Lower Bounds on Modular Counting: *Suppose there is an alternation-trading proof of*

$$\mathsf{NTIME}[n] \subseteq \mathsf{DTS}[n^c] \Longrightarrow \mathsf{DTS}[r(n)] \subseteq \mathsf{DTS}[s(n)]$$

for some constant c and polynomials $r(n), s(n)$. Then for all primes p, q with $p \neq q$ and for all $\varepsilon > 0$, there is a polynomial $s'_\varepsilon(n)$ such that $\lim_{\varepsilon \to 0} s'_\varepsilon(n) = s(n)$ and

$$\begin{aligned}(\mathsf{MOD}_p\mathsf{TIME}[n] \subseteq \mathsf{DTS}[n^{c-\varepsilon}] \text{ and } \mathsf{MOD}_q\mathsf{TIME}[n] \subseteq \mathsf{DTS}[n^{c-\varepsilon}]) \\ \Longrightarrow \quad \mathsf{DTS}[r(n)] \subseteq \mathsf{DTS}[s'_\varepsilon(n)].\end{aligned}$$

Proof Sketch of Transfer Principle. First, observe that there are concrete analogues to the three rules used in an alternation-trading proof:

1. (MOD Speedup) Theorem 1.1 says that

$$\begin{aligned}\mathsf{DTS}[T] &\subseteq (\mathsf{MOD}_p\ B \cdot n^{o(1)})(\mathsf{MOD}_q\ \log T)\mathsf{DTS}[T^{1+\varepsilon}/B] \\ \mathsf{DTS}[T] &\subseteq (\mathsf{MOD}_q\ B \cdot n^{o(1)})(\mathsf{MOD}_p\ \log T)\mathsf{DTS}[T^{1+\varepsilon}/B].\end{aligned}$$

   Moreover, the modular counting algorithm that simulates DTS reads only $n^{o(1)}$ bits guessed in the first MOD mode of the algorithm.

2. (MOD Removal) If $\mathsf{MOD}_p\mathsf{TIME}[n] \subseteq \mathsf{DTS}[n^c]$, $\mathsf{MOD}_q\mathsf{TIME}[n] \subseteq \mathsf{DTS}[n^c]$, then for $b \geq a \geq 1$,

$$\begin{aligned}(\mathsf{MOD}_p\ n^a)(\mathsf{MOD}_q\ n^b)\mathsf{DTS}[n^b] &\subseteq (\mathsf{MOD}_p\ n^a)\mathsf{DTS}[n^{bc}] \\ (\mathsf{MOD}_q\ n^a)(\mathsf{MOD}_p\ n^b)\mathsf{DTS}[n^b] &\subseteq (\mathsf{MOD}_q\ n^a)\mathsf{DTS}[n^{bc}],\end{aligned}$$

   which holds by standard padding arguments.

3. (MOD Combination) For all primes p, $(\mathsf{MOD}_p\ n^a)(\mathsf{MOD}_p\ n^b)\mathsf{DTS}[n^d] \subseteq (\mathsf{MOD}_p\ n^a + n^b)\mathsf{DTS}[n^d]$. This was shown by Beigel and Gill [BG92]; Papadimitriou and Zachos [PZ83] effectively proved the result for $p = 2$. The proof uses Fermat's Little Theorem that $a^{p-1} \equiv 1 \mod p$.

Let $\varepsilon > 0$. By assumption, there is a proof of

$$\mathsf{NTIME}[n] \subseteq \mathsf{DTS}[n^c] \implies \mathsf{DTS}[r(n)] \subseteq \mathsf{DTS}[s(n)]$$

that begins with the class $\mathsf{DTS}[r(n)]$ and applies the three rules (Quantifier Speedup, Quantifier Removal, and Quantifier Combination) in some sequence. Each such rule application places $\mathsf{DTS}[r(n)]$ inside of a new class, until $\mathsf{DTS}[r(n)]$ is placed in $\mathsf{DTS}[s(n)]$. To prove that

$$\begin{aligned}(\mathsf{MOD}_p\mathsf{TIME}[n] \subseteq \mathsf{DTS}[n^{c-\varepsilon}] \text{ and } \mathsf{MOD}_q\mathsf{TIME}[n] \subseteq \mathsf{DTS}[n^{c-\varepsilon}])& \\ \implies \quad \mathsf{DTS}[r(n)] \subseteq \mathsf{DTS}[s'_\varepsilon(n)]&\end{aligned}$$

for some $s'_\varepsilon(n)$, start with $\mathsf{DTS}[r(n)]$ and apply the modular analogues of the three rules (MOD Speedup, MOD Removal, and MOD Combination) in place of the original rules, deriving analogous containments.

In particular, suppose that every $\exists$-quantifier ($\forall$-quantifier) in the original proof is systematically replaced with a MOD_p mode (respectively, MOD_q mode) in the proof of $\mathsf{DTS}[r(n)] \subseteq \mathsf{DTS}[s'_\varepsilon(n)]$. The Quantifier Removal and Quantifier Combination rules have exactly the same parameters as the MOD Removal and MOD Combination rules, but each invocation of MOD Speedup (instead of Quantifier Speedup) in the new proof results in an additive ε factor in the exponent. However, the resulting polynomial runtime $s'_\varepsilon(n)$ of the final DTS class still has the property that if $\varepsilon = 0$, then the derived polynomial is exactly $s(n)$. Therefore $\lim_{\varepsilon\to 0} s'_\varepsilon(n) = s(n)$. □

6.2. Implications. The transfer principle implies two significant corollaries. Both say that certain proofs-by-contradiction that $\mathsf{NTIME}[n] \nsubseteq \mathsf{DTS}[n^c]$ can be mapped over to proofs that $\mathsf{MOD}_k\mathsf{TIME}[n] \nsubseteq \mathsf{DTS}[n^{c-\varepsilon}]$, for various integers k.

Corollary 6.2. *Suppose there is an alternation-trading proof of*

$$\mathsf{NTIME}[n] \subseteq \mathsf{DTS}[n^c] \Longrightarrow \mathsf{DTS}[n^{k+\delta}] \subseteq \mathsf{DTS}[n^k],$$

for some $k \geq 1$ and $\delta > 0$. Then for every $m > 1$ that is not a prime power, and for all $\varepsilon > 0$,

$$\mathsf{MOD}_m\mathsf{TIME}[n] \nsubseteq \mathsf{DTS}[n^{c-\varepsilon}].$$

First we show that if $\mathsf{NTIME}[n] \subseteq \mathsf{DTS}[n^c] \Longrightarrow \mathsf{DTS}[n^{k+\delta}] \subseteq \mathsf{DTS}[n^k]$, then actually the separation $\mathsf{NTIME}[n] \nsubseteq \mathsf{DTS}[n^c]$ holds. This result is necessary because we do not actually know if $\mathsf{DTS}[n^{k+\delta}] \nsubseteq \mathsf{DTS}[n^k]$ holds unconditionally (note the space bounds of both classes are the same).

THEOREM 6.3. *If* $\mathsf{NTIME}[n] \subseteq \mathsf{DTS}[n^c]$, *then* $\mathsf{DTS}[n^{k+\delta}] \not\subseteq \mathsf{DTS}[n^k]$ *for all* $k \geq 1$ *and* $\delta > 0$.

PROOF. Suppose that $\mathsf{NTIME}[n] \subseteq \mathsf{DTS}[n^c]$ and $\mathsf{DTS}[n^{k+\delta}] \subseteq \mathsf{DTS}[n^k]$ hold. By translation, $\mathsf{DTS}[n^{k+\delta}] \subseteq \mathsf{DTS}[n^k]$ implies

$$\mathsf{DTS}[n^{(k+\delta)^2/k}] \subseteq \mathsf{DTS}[n^{k+\delta}] \subseteq \mathsf{DTS}[n^k],$$

and therefore

$$\mathsf{DTS}[n^{(k+\delta)\cdot((k+\delta)/k)^i}] \subseteq \mathsf{DTS}[n^k],$$

for all $i \geq 0$. Since $\delta > 0$, this implies $\mathsf{DTS}[n^L] \subseteq \mathsf{DTS}[n^k]$ for all $L \geq k$. Thus $\mathsf{NTIME}[n] \subseteq \mathsf{DTS}[n^c]$ implies that for all $L > k$,

$$\mathsf{NTIME}[n^L] \subseteq \mathsf{DTS}[n^{Lc}] \subseteq \mathsf{DTS}[n^k] \subseteq \mathsf{coNTIME}[n^k],$$

which contradicts the fact that $\mathsf{NTIME}[n^L] \not\subseteq \mathsf{coNTIME}[n^{L-\varepsilon}]$ for all $\varepsilon > 0$ (this follows by a standard diagonalization argument). □

There is a modular analogue of this separation as well, with an identical proof:

THEOREM 6.4. *Let* $p \geq 2$ *be arbitrary. If* $\mathsf{MOD}_p\mathsf{TIME}[n] \subseteq \mathsf{DTS}[n^c]$ *then* $\mathsf{DTS}[n^{k+\delta}] \not\subseteq \mathsf{DTS}[n^k]$ *for all* $k \geq 1$ *and* $\delta > 0$.

Using the above, we can obtain the Corollary.

PROOF OF COROLLARY 6.2. If m is not a prime power, then it has two distinct primes p and q as factors. Therefore, $\mathsf{MOD}_p\mathsf{TIME}[t] \subseteq \mathsf{MOD}_m\mathsf{TIME}[t]$ and $\mathsf{MOD}_q\mathsf{TIME}[t] \subseteq \mathsf{MOD}_m\mathsf{TIME}[t]$ for all time bounds t (cf. [BG92] for a proof). Thus $\mathsf{MOD}_m\mathsf{TIME}[n] \subseteq \mathsf{DTS}[n^{c-\varepsilon}]$ implies

$$\mathsf{MOD}_p\mathsf{TIME}[n] \subseteq \mathsf{DTS}[n^{c-\varepsilon}] \text{ and}$$

$$\mathsf{MOD}_q\mathsf{TIME}[n] \subseteq \mathsf{DTS}[n^{c-\varepsilon}].$$

Therefore, by the transfer principle one can use the presumed alternation-trading proof to obtain $\mathsf{DTS}[n^{k+\delta}] \subseteq \mathsf{DTS}[s'_\varepsilon(n)]$, where $s'_\varepsilon(n)$ is a polynomial that converges to n^k for $\varepsilon \to 0$, contradicting Theorem 6.4. □

COROLLARY 6.5. *Suppose there is an alternation-trading proof of*

$$\mathsf{NTIME}[n] \subseteq \mathsf{DTS}[n^c] \Longrightarrow \mathsf{DTS}[n^{k+\delta}] \subseteq \mathsf{DTS}[n^k],$$

for some $k \geq 1$ and $\delta > 0$. Then for all primes p except for possibly one of them, $\mathsf{MOD}_p\mathsf{TIME}[n] \not\subseteq \mathsf{DTS}[n^{c-\varepsilon}]$ for all $\varepsilon > 0$. That is, there is at most one prime p for which $\mathsf{MOD}_p\mathsf{TIME}[n]$ is in $n^{c-\varepsilon}$ time and $n^{o(1)}$ space.

PROOF. If there are two different primes p and q with $\mathsf{MOD}_p\mathsf{TIME}[n] \subseteq \mathsf{DTS}[n^{c-\varepsilon}]$ and $\mathsf{MOD}_q\mathsf{TIME}[n] \subseteq \mathsf{DTS}[n^{c-\varepsilon}]$, then the transfer principle applies as in Corollary 6.2, and a contradiction can be established with Theorem 6.4. Thus at most one prime p can satisfy $\mathsf{MOD}_p\mathsf{TIME}[n] \subseteq \mathsf{DTS}[n^{c-\varepsilon}]$. □

6.3. New Time Lower Bound for SAT. We now proceed with a new time-space lower bound for $\mathsf{NTIME}[n]$. Theorem 1.2 follows immediately from it, coupled with the above two results and the completeness results for SAT and MOD-SAT (Theorems 2.1 and 2.2).

THEOREM 6.6. *For all $c \geq 1$ such that $c^3 - c^2 - 2c + 1 < 0$,* i.e. *$c < 2\cos(\pi/7)$,*

$$\mathsf{NTIME}[n] \not\subseteq \mathsf{DTS}[n^c].$$

Furthermore, the result can be obtained with an alternation-trading proof that $\mathsf{NTIME}[n] \subseteq \mathsf{DTS}[n^c]$ implies $\mathsf{DTS}[n^{k+\delta}] \subseteq \mathsf{DTS}[n^k]$ for some $k \geq 1$ and $\delta > 0$.

For the sake of mathematical curiosity, let us first make a few remarks about the constant $2\cos(\pi/7) = 1.8019\cdots$ and how it arises. Whereas the golden ratio is the unique solution in $(1, 2)$ to the equation $c^2 = 1 + 1/(c-1)$, our quantity is the unique solution in $(1, 2)$ to $c^2 = 2 + 1/(c-1)$.

To compute the roots of $p(c) = c^3 - c^2 - 2c + 1$, with a little foresight we let $c = 2\cos(u)$, and recall that $2\cos(x) = e^{ix} + e^{-ix}$. Plugging into p, we get

$$\begin{aligned} p(c) &= (e^{iu} + e^{-iu})^3 - (e^{iu} + e^{-iu})^2 - 2(e^{iu} + e^{-iu}) + 1 \\ &= (e^{3iu} + e^{-3iu}) - (e^{2iu} + e^{-2iu}) + (e^{iu} + e^{-iu}) - 1, \end{aligned}$$

after simplifying. Multiplying by e^{3iu}, the above sum becomes

$$(e^{6iu} + 1) - (e^{5iu} + e^{iu}) + (e^{4iu} + e^{2iu}) - e^{3iu},$$

which is

$$e^{6iu} - e^{5iu} + e^{4iu} - e^{3iu} + e^{2iu} - e^{iu} + 1 = \frac{e^{7iu} + 1}{e^{iu} + 1}.$$

For $u \neq \pi$, the roots of p are given by $e^{7iu} = -1$, leading to the three equations:

$$e^{7u_1 i} = e^{\pi i},\ e^{7u_2 i} = e^{3\pi i},\ \text{and}\ e^{7u_3 i} = e^{5\pi i}.$$

That is, the roots are $c_1 = 2\cos(\pi/7), c_2 = 2\cos(3\pi/7), c_3 = 2\cos(5\pi/7)$, where only $c_1 > 1$.

The lower bound proof requires a speedup simulation from our previous work [Wil06]. Effectively, this simulation says that if we assume that nondeterministic linear time can be simulated in n^c time and $n^{o(1)}$ space, that assumption can be applied to improve the special case of Nepomnjascii's theorem, proven earlier in Theorem 5.1.

LEMMA 6.7 (Conditional Speedup [Wil06]). *Suppose there is a $c < 2$ satisfying $\mathsf{NTIME}[n] \subseteq \mathsf{DTS}[n^c]$. Then for all $d < c/(c-1)$,*

$$\mathsf{DTS}[n^d] \subseteq (\exists\ n)(\forall\ \log n)\mathsf{DTS}[n] \cap (\forall\ n)(\exists\ \log n)\mathsf{DTS}[n].$$

Moreover, the proof uses only the three rules of alternation-trading proofs.

Note that $c < 2$ implies $c/(c-1) > 2$, so the above alternating speedup of n^d time and $n^{o(1)}$ space is better than the square root gotten from Theorem 5.1.

PROOF. Define the sequence $d_0 := 2$, $d_k := 1 + \frac{d_{k-1}}{c}$. We prove by induction that for all $k \geq 0$,

$$\mathsf{DTS}[n^{d_k}] \subseteq (\exists\ n)(\forall\ \log n)\mathsf{DTS}[n] \cap (\forall\ n)(\exists\ \log n)\mathsf{DTS}[n].$$

As the sequence $\{d_k\}$ is monotone increasing and converges to $c/(c-1)$ (when $c < 2$), the theorem follows.

The case $k = 0$ follows from Theorem 5.1, as $\mathsf{DTS}[n^2] \subseteq \Sigma_2\mathsf{TIME}[n^{1+o(1)}] \cap \Pi_2\mathsf{TIME}[n^{1+o(1)}]$, using the Quantifier Speedup rule of alternation-trading proofs.

For the inductive step, consider the class $\mathsf{DTS}[n^{d_{k+1}}] = \mathsf{DTS}[n^{1+d_k/c}]$. By the Quantifier Speedup rule, $\mathsf{DTS}[n^{1+d_k/c}]$ is contained in

$$(\exists\ n)(\forall\ \log n)\mathsf{DTS}[n^{d_k/c}].$$

Note that $d_k/c \geq 1$. The conondeterministic part of the Σ_2 class above always has input of length $O(n)$. Therefore by the Quantifier Removal rule, we can apply the assumption $\mathsf{NTIME}[n] \subseteq \mathsf{DTS}[n^c]$ to the $(\forall\ \log n)\mathsf{DTS}[n^{d_k/c}]$ part, obtaining the class

$$(\exists\ n)\mathsf{DTS}[n^{d_k}].$$

Finally, by induction hypothesis, $\mathsf{DTS}[n^{d_k}] \subseteq (\exists n)(\forall \log n)\mathsf{DTS}[n]$, so the above class is contained in

$$(\exists\ n)(\exists\ n)(\forall \log n)\mathsf{DTS}[n] \subseteq (\exists\ n)(\forall\ \log n)\mathsf{DTS}[n],$$

by the Quantifier Combination rule. By closure of DTS under complementation, we have that $\mathsf{DTS}[n^{d_{k+1}}] \subseteq (\forall\ n)(\exists\ \log n)\mathsf{DTS}[n]$ as well. □

The crux of the $n^{1.801}$ lower bound rests in the following result, which is a subtle combination of the golden ratio proof strategy of Fortnow-Van Melkebeek and our Conditional Speedup Lemma (Lemma 6.7).

LEMMA 6.8. *Suppose there is $c < 2$ such that $\mathsf{NTIME}[n] \subseteq \mathsf{DTS}[n^c]$. Then for all integers $k \geq 1$, and d satisfying $c^2 \geq d$, $c \leq d < c/(c-1)$,*

$$\mathsf{DTS}[n^{d+\sum_{i=1}^{k}(c^2/d)^i}] \subseteq \Sigma_2\mathsf{TIME}[n^{(c^2/d)^k+o(1)}] \cap \Pi_2\mathsf{TIME}[n^{(c^2/d)^k+o(1)}].$$

The proof applies only the three rules of alternation-trading proofs.

Let us say a few words about why this lemma is helpful. Our goal is to have $\mathsf{DTS}[n^a] \subseteq \Sigma_2\mathsf{TIME}[n^b]$ where the ratio a/b is as large as possible. Then, using Quantifier Removal, we may prove consequences of the form

$$\mathsf{DTS}[n^a] \subseteq \Sigma_2\mathsf{TIME}[n^b] \subseteq \mathsf{NTIME}[n^{bc}] \subseteq \mathsf{DTS}[n^{bc^2}].$$

For $a/b > c^2$, we can get a contradiction with Theorem 6.3. The Conditional Speedup Lemma gives us an a/b ratio of $c/(c-1) - \varepsilon$. To get a larger ratio, we let a and b increase, and doing so we can exceed $c/(c-1) - \varepsilon$. The trick is interleave applications of the Conditional Speedup Lemma (where the alternating simulations always choose a number of configurations that is linear in the input length) with Quantifier Speedups where we choose more than just a linear number of configurations.

PROOF OF LEMMA 6.8. Suppose $c < 2$ is such that $\mathsf{NTIME}[n] \subseteq \mathsf{DTS}[n^c]$. Pick d satisfying the required properties. The proof is by induction on k. For $k = 0$, the task is just to show $\mathsf{DTS}[n^d] \subseteq \Sigma_2\mathsf{TIME}[n^{1+o(1)}]$, which is precisely Lemma 6.7.

For the inductive step, consider the class $\mathsf{DTS}[n^{d+\sum_{i=1}^{k}(c^2/d)^i}]$. By Theorem 5.1,

$$\mathsf{DTS}[n^{d+\sum_{i=1}^{k}(c^2/d)^i}] \subseteq (\exists\ n^{(c^2/d)^k})(\forall\ \log n)\mathsf{DTS}[n^{d+\sum_{i=1}^{k-1}(c^2/d)^i}],$$

where the $\mathsf{DTS}[\cdots]$ part of the Σ_2 computation has input of length $n + n^{o(1)}$ (the original input x, and two configurations). This step uses the Quantifier Speedup rule. By the induction hypothesis,

$$\begin{aligned} & (\exists\ n^{(c^2/d)^k})(\forall\ \log n)\mathsf{DTS}[n^{d+\sum_{i=1}^{k-1}(c^2/d)^i}] \\ \subseteq\ & (\exists\ n^{(c^2/d)^k})(\forall\ \log n)\Pi_2\mathsf{TIME}[n^{(c^2/d)^{k-1}+o(1)}]. \end{aligned}$$

Applying the assumption that $\mathsf{NTIME}[n] \subseteq \mathsf{DTS}[n^c]$ (via the Quantifier Removal rule) to the Π_2 part (which takes input of length $n + n^{o(1)}$),

$$\begin{aligned} & (\exists\ n^{(c^2/d)^k})(\forall\ \log n)\Pi_2\mathsf{TIME}[n^{(c^2/d)^{k-1}+o(1)}] \\ \subseteq\ & (\exists\ n^{(c^2/d)^k})(\forall\ \log n)\mathsf{coNTIME}[n^{c\cdot(c^2/d)^{k-1}+o(1)}]. \end{aligned}$$

By the Quantifier Combination rule,

$$\begin{aligned} & (\exists\ n^{(c^2/d)^k})(\forall\ \log n)\mathsf{coNTIME}[n^{c\cdot(c^2/d)^{k-1}+o(1)}] \\ \subseteq\ & (\exists\ n^{(c^2/d)^k})\mathsf{coNTIME}[n^{c\cdot(c^2/d)^{k-1}+o(1)}]. \end{aligned}$$

Now the input to the $\mathsf{coNTIME}$ part has length $O(n^{(c^2/d)^k}) \leq O(n^{c\cdot(c^2/d)^{k-1}})$, since $d \geq c$. Therefore we can use Quantifier Removal again to obtain

$$(\exists\ n^{(c^2/d)^k})\mathsf{coNTIME}[n^{c\cdot(c^2/d)^{k-1}+o(1)}] \subseteq (\exists\ n^{(c^2/d)^k})\mathsf{DTS}[n^{c^2\cdot(c^2/d)^{k-1}}].$$

But by Lemma 6.7, the DTS part of the above can be replaced with a Σ_2 computation, in particular

$$(\exists\ n^{(c^2/d)^k})\mathsf{DTS}[n^{c^2\cdot(c^2/d)^{k-1}}] \subseteq (\exists\ n^{(c^2/d)^k})(\exists\ n^{(c^2/d)^k})(\forall\ \log n)\mathsf{DTS}[n^{(c^2/d)^k}].$$

Finally,

$$(\exists\ n^{(c^2/d)^k})(\exists\ n^{(c^2/d)^k})(\forall\ \log n)\mathsf{DTS}[n^{(c^2/d)^k}] \subseteq (\exists\ n^{(c^2/d)^k})(\forall\ \log n)\mathsf{DTS}[n^{(c^2/d)^k}]$$

by Quantifier Combination. This completes the proof. □

Now we show how Lemma 6.8 implies Theorem 6.6, the $n^{1.801}$ lower bound.

Proof of Theorem 6.6. Assuming $\mathsf{NTIME}[n] \subseteq \mathsf{DTS}[n^c]$ and Lemma 6.8,

$$\begin{aligned} \mathsf{DTS}[n^{d+\sum_{i=1}^{k}(c^2/d)^i}] &\subseteq \Sigma_2\mathsf{TIME}[n^{(c^2/d)^k+o(1)}] \\ &\subseteq \mathsf{NTIME}[n^{c(c^2/d)^k+o(1)}] \subseteq \mathsf{DTS}[n^{c^2(c^2/d)^k}]. \end{aligned}$$

A contradiction with Theorem 6.3 is reached (and thus $\mathsf{NTIME}[n] \not\subseteq \mathsf{DTS}[n^c]$) precisely when

$$d + \sum_{i=1}^{k} (c^2/d)^i > c^2 \cdot (c^2/d)^k,$$

that is, when

$$\begin{aligned} c^2 < \sum_{i=1}^{k} \left(\frac{c^2}{d}\right)^{i-k} + d \cdot \left(\frac{d}{c^2}\right)^k \quad &\iff \quad c^2 < \sum_{j=0}^{k-1} \left(\frac{d}{c^2}\right)^j + d \cdot \left(\frac{d}{c^2}\right)^k \\ &\iff \quad c^2 < \frac{1 - \left(\frac{d}{c^2}\right)^k}{1 - \left(\frac{d}{c^2}\right)} + d \cdot \left(\frac{d}{c^2}\right)^k \end{aligned} \tag{6.9}$$

By assumption, $(d/c^2)^k < 1$ for all $k \geq 1$, and

$$\lim_{k \to \infty} (d/c^2)^k = 0.$$

Therefore, assuming $c \geq \phi$ (the golden ratio), for any $\varepsilon > 0$ we can set $d = c/(c-1) - \varepsilon$ and find a k such that $0 < (d/c^2)^k \leq \varepsilon$. Hence the inequality (6.9) is implied by the inequality:

$$c^2 < \frac{1 - \varepsilon}{1 - \frac{c/(c-1) - \varepsilon}{c^2}}.$$

Simple algebraic manipulation yields the equivalent condition

$$c^2 - (c/(c-1) - \varepsilon) < (1 - \varepsilon).$$

Multiplying through by $(c-1)$, the condition becomes

$$\begin{aligned} c^2(c-1) - (c - \varepsilon(c-1)) &< ((c-1) - \varepsilon(c-1)) \\ \iff c^3 - c^2 - 2c + 1 &< 2\varepsilon(1-c) \end{aligned} \tag{6.10}$$

Now, as ε approaches 0, the RHS approaches 0. We arrive at the following condition implying a contradiction:

$$c^3 - c^2 - 2c + 1 < 0,$$

which is what we wanted to prove. That is, for any c satisfying $c^3 - c^2 - 2c + 1 < 0$, one can choose an $\varepsilon > 0$ such that c and ε satisfy inequality (6.10), and therefore inequality (6.9) as well. □

The above argument can be extended to prove the following time-space tradeoff for SAT:

COROLLARY 6.11. *For all $c < 2\cos(\pi/7)$ there is a $d \in (0,1)$ such that* SAT *is not in* $\mathsf{DTISP}[n^c, n^d]$.

PROOF. (Sketch) In the above proofs, one can replace the $n^{o(1)}$ space bound by n^d for a sufficiently small $d > 0$. The sizes of quantifiers in the alternating simulations increase only by an additive factor of $q_k d$ in the exponents, where q_k is a constant that depends on the (finite) sequence of k Quantifier Speedup and Quantifier Removal rules used. □

7. Lower Bounds for Counting Solutions Modulo Particular Primes

In the previous section, we obtained a time lower bound for MOD_p-SAT for all but possibly one prime p; however, the result was non-constructive. Such a result is frustrating, as it does not let us point to a single prime for which the lower bound actually holds. We can prove lower bounds for counting satisfying assignments modulo *particular* primes, provided that a conjectured time hierarchy theorem holds. We recall Hypothesis 1.3 and Theorem 1.4 from the Introduction:

HYPOTHESIS 1.3. *For all primes p, there exists a prime $q \neq p$ and time constructible $T(n) \geq n^2$ such that $\mathsf{MOD}_p\mathsf{TIME}[T] \not\subseteq \mathsf{MOD}_q\mathsf{TIME}[T^{1-\varepsilon}]$, for all $\varepsilon > 0$.*

THEOREM 1.4. *If Hypothesis 1.3 holds for a given prime p, then MOD_p-SAT requires $\Omega(n^c)$ time on $n^{o(1)}$ space machines, for $c < \phi$, the golden ratio.*

To prove the theorem, we require a modular version of the Conditional Speedup Theorem (Theorem 6.7):

THEOREM 7.1. *If there is a $c < 2$ and prime p satisfying $\mathsf{MOD}_p\mathsf{TIME}[n] \subseteq \mathsf{DTS}[n^c]$, then for all $d < c/(c-1)$ and primes $q \neq p$,*

$$\mathsf{DTS}[n^d] \subseteq (\mathsf{MOD}_q\ n)(\mathsf{MOD}_p\ \log n)\mathsf{DTS}[n].$$

PROOF. Analogous to the proof of Theorem 6.7. □

Now we are prepared to prove Theorem 1.4. Suppose for contradiction that

$$\mathsf{MOD}_p\mathsf{TIME}[n] \subseteq \mathsf{DTS}[n^c] \tag{7.2}$$

and for some $T(n) \geq n^2$,

$$\mathsf{MOD}_p\mathsf{TIME}[T] \not\subseteq \mathsf{MOD}_q\mathsf{TIME}[T^{1-\varepsilon}]. \tag{7.3}$$

Starting with the class $\mathsf{MOD}_p\mathsf{TIME}[n^{1/(c-1)-\varepsilon}]$, we derive

$$\begin{aligned}
\mathsf{MOD}_p\mathsf{TIME}[n^{1/(c-1)-\varepsilon}] &\subseteq \mathsf{DTS}[n^{c/(c-1)-c\varepsilon}] \text{ by (7.2)} \\
&\subseteq (\mathsf{MOD}_q\ n)(\mathsf{MOD}_p\ \log n)\mathsf{DTS}[n] \text{ by Theorem 7.1} \\
&\subseteq (\mathsf{MOD}_q\ n)\mathsf{DTS}[n^c] \text{ by (7.2)} \\
&\subseteq \mathsf{MOD}_q\mathsf{TIME}[n^c].
\end{aligned}$$

Suppose $c(c-1) < 1$. Then the above inclusion implies $\mathsf{MOD}_p\mathsf{TIME}[n^2] \subseteq \mathsf{MOD}_q\mathsf{TIME}[n^{2-\delta}]$ for some $\delta > 0$, but this contradicts (7.3). Therefore (7.2) does not hold for any c satisfying $c(c-1) < 1$, *i.e.* for c less than the golden ratio $\phi \approx 1.618$. This completes the proof of Theorem 1.4.

8. Deterministic Small Space Versus Modular Counting

Using the Speedup By Modular Counting Theorem, one can establish other "modular-counting analogues" of known results about the power of alternating computation. For example, if $\mathsf{MOD}_6\mathsf{TIME}[n^k]$ is efficiently closed under Turing reductions for any k, then a major separation result would be obtained. Recall that $\mathsf{SC} := \mathsf{DTISP}[n^{O(1)}, (\log n)^{O(1)}]$.

THEOREM 8.1. *If there is some $k \geq 1$ such that $(\mathsf{MOD}_6\mathsf{MOD}_6)\mathsf{TIME}[n^k] \subseteq \mathsf{MOD}_6\mathsf{TIME}[n^{k+o(1)}]$, then $\mathsf{SC} \neq \mathsf{MOD}_6\mathsf{P}$.*

Therefore, if a modular-alternating machine with two MOD_6 modes can be efficiently simulated by a machine with only one MOD_6 mode, then it follows that logarithmic space is different from $\mathsf{MOD}_6\mathsf{P}$. Theorem 8.1 is a "modular analogue" of a result for nondeterminism that follows from the work of Fortnow [For00].

THEOREM 8.2. *If $\Sigma_2\mathsf{TIME}[n^k] \subseteq \mathsf{NTIME}[n^{k+o(1)}]$ then $\mathsf{SC} \neq \mathsf{NP}$.*

PROOF OF THEOREM 8.1. It follows from the Speedup by Modular Counting Theorem that for any integer $\ell \geq 1$ and real $k \geq 1$, $\mathsf{DTISP}[n^{\ell+k}, \mathrm{poly}(\log n)]$ can be simulated by a machine with signature $(\mathsf{MOD}_6)^{\ell+1}$ (*i.e.* $\ell+1$ separate MOD_6 modes) that guesses $O(n)$ bits in each mode, and the last deterministic mode runs in $n^{k+\varepsilon}$ time, for any $\varepsilon > 0$. By the assumption, the last ℓ MOD_6 modes of this machine can be "removed", while only increasing the runtime by a $o(1)$ additive factor in the exponent. Therefore

$$\mathsf{DTISP}[n^{\ell+k}, \mathrm{poly}(\log n)] \subseteq \mathsf{MOD}_6\mathsf{TIME}[n^{k+o(1)}] \tag{8.3}$$

for any $\ell \geq 1$. Now, if $\mathsf{SC} = \mathsf{MOD}_6\mathsf{P}$, then MOD_6-SAT $\in \mathsf{DTISP}[n^j, \mathrm{poly}(\log n)]$, for some j. By a slightly strengthened version of Theorem 2.2 (cf. [vM07]), we also have that $\mathsf{MOD}_6\mathsf{TIME}[n] \subseteq \mathsf{DTISP}[n^{j+o(1)}, \mathrm{poly}(\log n)]$. But this implies

$$\mathsf{MOD}_6\mathsf{TIME}[n^{1+k}] \subseteq \mathsf{DTISP}[n^{j(1+k)+o(1)}, \mathrm{poly}(\log n)] \subseteq \mathsf{MOD}_6\mathsf{TIME}[n^{k+o(1)}],$$

by setting $\ell = (j(1+k) - k)$ and applying inclusion (8.3). This contradicts the time hierarchy for MOD_6 computations, so we must conclude that $\mathsf{SC} \neq \mathsf{MOD}_6\mathsf{P}$. □

Similarly, the following can be proved with essentially the same argument.

THEOREM 8.4. *If there are primes $p \neq q$ and $k \geq 1$ such that $\mathsf{MOD}_p\mathsf{TIME}[n^k] \subseteq \mathsf{MOD}_q\mathsf{TIME}[n^{k+o(1)}]$ and $\mathsf{MOD}_q\mathsf{TIME}[n^k] \subseteq \mathsf{MOD}_p\mathsf{TIME}[n^{k+o(1)}]$, then $\mathsf{SC} \neq \mathsf{MOD}_p\mathsf{P}$ and $\mathsf{SC} \neq \mathsf{MOD}_q\mathsf{P}$.*

PROOF. (Sketch) Another corollary of the Speedup by Modular Counting Theorem is that $\mathsf{DTISP}[n^{\ell+k}, \mathrm{poly}(\log n)]$ can be simulated by a machine with signature $(\mathsf{MOD}_p\ \mathsf{MOD}_q)^{\ell+1}$ that runs in $n^{1+o(1)}$ time in each modular mode, and the last deterministic mode runs in $n^{k+\varepsilon}$ time, for arbitrarily small ε. If either $\mathsf{SC} = \mathsf{MOD}_p\mathsf{P}$ or $\mathsf{SC} = \mathsf{MOD}_q\mathsf{P}$ were to hold, we could obtain a contradiction by applying the two assumptions, just as in the proof of Theorem 8.1. □

Finally, we give an unconditional separation. Fortnow [For00] showed that NL is not equal to any non-constant level of the polynomial time hierarchy. We can give an analogous separation result for the class SC. For an integer m and function $s(n)$, define the class $(\mathsf{MOD}_m)^{s(n)}\mathsf{P}$ to contain just those languages recognized by a modular-alternating machine that runs in polynomial time, uses only MOD_m modes, and makes at most $s(n)$ mode alternations in any of its computation paths.

THEOREM 8.5. *Let $s(n)$ be any monotone increasing unbounded function. Then $\mathsf{SC} \neq (\mathsf{MOD}_m)^{s(n)}\mathsf{P}$, for any composite m that is not a prime power.*

PROOF. (Sketch) We claim that for all $\varepsilon > 0$,

$$\mathsf{SC} \subseteq \bigcup_{k \geq 1} (\mathsf{MOD}_m\ n)^k \mathsf{DTIME}[n^{1+\varepsilon}],$$

where $(\mathsf{MOD}_m\ n)^a\mathcal{C}$ is shorthand for the class

$$\underbrace{(\mathsf{MOD}_m\ n) \cdots (\mathsf{MOD}_m\ n)}_{a} \mathcal{C}.$$

From the proof of Theorem 8.1, it follows that for any $k \geq 1$ and sufficiently small $\varepsilon > 0$,

$$\mathsf{DTISP}[n^k, \mathrm{poly}(\log n)] \subseteq (\mathsf{MOD}_m\ n)^{k+1}\mathsf{DTISP}[n^{1+\varepsilon}, \mathrm{poly}(\log n)].$$

This proves the claim. The theorem follows from the fact that

$$\bigcup_{k \geq 0} (\mathsf{MOD}_m\ n)^k \mathsf{DTIME}[n^{1+\varepsilon}] \subsetneq (\mathsf{MOD}_m)^{s(n)}\mathsf{P},$$

which holds by a simple diagonalization. □

9. Conclusion

We have proven the first superlinear time lower bounds for counting NP solutions modulo small integers, on random access machines that use subpolynomial space. For example, the best known time-space lower bound for MOD_6-SAT is the same as that for SAT. To arrive at our results, we discovered a way to transfer lower bound proofs for nondeterminism to the setting of modular counting, using a lemma that shows the number of configuration sequences of a canonical machine with a prescribed number of mistakes on a given input depends solely upon the space usage of the machine and the acceptance/rejection condition of that input. Such a tool may prove to be useful in other time-space lower bound arguments as well. For example, one may be able to prove strong time-space lower bounds for the MAJORITY SAT problem using the Counting Lemma, but we have not yet found a way to do this. Currently the best time-space lower bound we know for MAJORITY SAT is the same as the one known for SAT.

An $\Omega(n^{\phi - o(1)})$ time and $n^{o(1)}$ space lower bound for MOD_p-SAT for a *particular* prime p follows from a conjectured time hierarchy, which we consider to be an interesting open problem:

Conjecture: There are primes $q \neq p$ and time constructible $T(n) \geq n^2$ such that for all $\varepsilon > 0$,

$$\mathsf{MOD}_p\mathsf{TIME}[T] \not\subseteq \mathsf{MOD}_q\mathsf{TIME}[T^{1-\varepsilon}].$$

We have recently formalized the "indirect diagonalization" paradigm behind ours and the previous lower bound arguments, and have implemented a computer program that can feasibly search over the space of short lower bound proofs that follow the alternation-trading rules [Wil07a, Wil07b]. Contrary to our expectations, preliminary results strongly suggest that the $\Omega(n^{2\cos(\pi/7)-\varepsilon})$ time lower bound is the best possible, using the current techniques. Therefore, to obtain better lower bounds in the future, it appears that one will need to add truly new ideas to the current arguments.

Acknowledgements

A preliminary version of this paper was a co-winner of the Ronald V. Book Best Student Paper Award at CCC'07. The author is grateful to Virginia Vassilevska, Dieter van Melkebeek, Ryan O'Donnell, Steven Rudich, and the anonymous referees for their very helpful comments which improved the exposition tremendously. In particular, the anonymous referees for the journal version discovered several simplifications that tightened the presentation of the main result.

References

[AG94] E. Allender and V. Gore. A uniform circuit lower bound for the permanent. *SIAM Journal on Computing* 23:1026–1049, 1994.

[All99] E. Allender. The permanent requires large uniform threshold circuits. *Chicago Journal of Theoretical Computer Science*, article 7, 1999.

[AKRRV01] E. Allender, M. Koucky, D. Ronneburger, S. Roy, and V. Vinay. Time-space tradeoffs in the counting hierarchy. In *Proceedings of IEEE Conference on Computational Complexity (CCC)*, 295–302, 2001.

[BST90] D. Barrington, H. Straubing, and D. Therien. Non-uniform automata over groups. *Information and Computation* 89(2):109–132, 1990.

[BBF98] R. Beigel, H. Buhrman, and L. Fortnow. NP might not be as easy as detecting unique solutions. In *Proceedings of ACM Symposium on Theory of Computing (STOC)*, 203–208, 1998.

[BG92] R. Beigel and J. Gill. Counting classes: thresholds, parity, mods, and fewness. *Theoretical Computer Science* 103(1):3–23, 1992.

[Ben89] C. H. Bennett. Time-space tradeoffs for reversible computation. *SIAM Journal on Computing* 18:766–776, 1989.

[CH90] J.-Y. Cai and L. A. Hemachandra. On the power of parity polynomial time. *Theory of Computing Systems* 23(1):95–106, 1990.

[CC06] J.-Y. Cai and V. Choudhary. Some results on matchgates and holographic algorithms. In *Proceedings of ICALP Vol. 1*, Springer-Verlag LNCS, 703–714, 2006.

[CL07a] J.-Y. Cai and P. Lu. Holographic algorithms: from art to science. In *Proceedings of ACM Symposium on Theory of Computing (STOC)*, 401–410, 2007.

[CL07b] J.-Y. Cai and P. Lu. Bases collapse in holographic algorithms. In *Proceedings of IEEE Conference on Computational Complexity (CCC)*, 292–304, 2007.

[CMTV98] H. Caussinus, P. McKenzie, D. Therien and H. Vollmer. Nondeterministic NC^1 computation. *Journal of Computer and System Sciences* 57(2):200–212, 1998.

[CGPT06] A. Chattopadhyay, N. Goyal, P. Pudlak, and D. Therien. Lower bounds for circuits with MOD_m gates. In *Proceedings of IEEE Symposium on Foundations of Computer Science (FOCS)*, 2006.

[DvM06] S. Diehl and D. Van Melkebeek. Time-space lower bounds for the polynomial-time hierarchy on randomized machines. *SIAM Journal on Computing* 36: 563-594, 2006.

[For00] L. Fortnow. Time-space tradeoffs for satisfiability. *J. Comput. Syst. Sci.* 60(2):337–353, 2000.

[FLVV05] L. Fortnow, R. Lipton, D. Van Melkebeek, and A. Viglas. Time-space lower bounds for satisfiability. *Journal of the ACM* 52(6):835–865, 2005.

[FvM00] L. Fortnow and D. Van Melkebeek. Time-space tradeoffs for nondeterministic computation. In *Proceedings of IEEE Conference on Computational Complexity (CCC)*, 2–13, 2000.

[Gro01] V. Grolmusz. A degree-decreasing lemma for (MOD-q - MOD-p) circuits. *Discrete Mathematics and Theoretical Computer Science* 4(2):247–254, 2001.

[Gup98] S. Gupta. Isolating an odd number of elements and applications in complexity theory. *Theory of Computing Systems* 31:27–40, 1998.

[Kan84] R. Kannan. Towards separating nondeterminism from determinism. *Mathematical Systems Theory* 17(1):29–45, 1984.

[KvM02] A. Klivans and D. Van Melkebeek. Graph nonisomorphism has subexponential size proofs unless the polynomial-time hierarchy collapses. *SIAM Journal on Computing* 31:1501–1526, 2002.

[LS90] R. Y. Levine and A. T. Sherman. A note on Bennett's time-space tradeoff for reversible computation. *SIAM Journal on Computing* 19(4):673–677, 1990.

[LV99] R. J. Lipton and A. Viglas. On the complexity of SAT. In *Proceedings of IEEE Symposium on Foundations of Computer Science (FOCS)*, 459-464, 1999.

[MS87] W. Maass and A. Schorr. Speed-up of Turing machines with one work tape and a two-way input tape. *SIAM Journal on Computing* 16(1):195–202, 1987.

[vM04] D. Van Melkebeek. Time-space lower bounds for NP-complete problems. In G. Paun, G. Rozenberg, and A. Salomaa (eds.), *Current Trends in Theor. Comp. Sci.* 265–291, World Scientific, 2004.

[vMR05] D. Van Melkebeek and R. Raz. A time lower bound for satisfiability. *Theoretical Computer Science* 348(2-3):311–320, 2005.

[vM07] D. Van Melkebeek. A survey of lower bounds for satisfiability and related problems. To appear in *Foundations and Trends in Theoretical Computer Science*, 2007.

[NRS95] A. V. Naik, K. W. Regan, D. Sivakumar. On quasilinear-time complexity theory. *Theoretical Computer Science* 148:325–349, 1995.

[Nep70] V. Nepomnjascii. Rudimentary predicates and Turing calculations. *Soviet Math. Doklady* 11:1462–1465, 1970.

[PZ83] C. H. Papadimitriou and S. Zachos. Two remarks on the power of counting. *Theoretical Computer Science 1983*, Springer LNCS 145:269–276, 1983.

[Smo87] R. Smolensky. Algebraic methods in the theory of lower bounds for Boolean circuit complexity. In *Proceedings of ACM Symposium on Theory of Computing (STOC)*, 77–82, 1987.

[TO92] S. Toda and M. Ogiwara. Counting classes are at least as hard as the polynomial-time hierarchy. *SIAM Journal on Computing* 21(2):316–328, 1992.

[Tou01] I. Tourlakis. Time-space tradeoffs for SAT on nonuniform machines. *Journal of Computer and System Sciences* 63(2):268–287, 2001.

[VV86] L. Valiant and V. Vazirani. NP is as easy as detecting unique solutions. *Theoretical Computer Science* 47(1):85–93, 1986.

[Val04] L. Valiant. Holographic algorithms. In *Proceedings of IEEE Symposium on Foundations of Computer Science (FOCS)*, 306–315, 2004.

[Val06] L. Valiant. Accidental algorithms. In *Proceedings of IEEE Symposium on Foundations of Computer Science (FOCS)*, 509–517, 2006.

[Wil06] R. Williams. Inductive time-space lower bounds for SAT and related problems. *Computational Complexity* 15:433–470, 2006.

[Wil07a] R. Williams. Algorithms and resource requirements for fundamental problems. Ph.D. Thesis, Carnegie Mellon University, CMU-CS-07-147, August 2007.

[Wil07b] R. Williams. Automated proofs of time lower bounds. Manuscript, 2007.

[XZZ07] M. Xia, P. Zhang, and W. Zhao. Computational complexity of counting problems on 3-regular planar graphs. *Theoretical Computer Science* 384(1):111–125, 2007.

Manuscript received August 19, 2007

R. Ryan Williams
Computer Science Department
Carnegie Mellon University
Pittsburgh, PA 15213
ryanw@cs.cmu.edu